NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A DCT-BASED IMAGE WATERMARKING ALGORITHM ROBUST TO CROPPING AND COMPRESSION.

by

Ioannis Retsas

March 2002

Co-advisors:	Ron J. Pieper
	Roberto Cristi
Second Reader:	David C. Jenn

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March, 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: A DCT-Based Image Watermarking Algorithm Robust To Cropping and Compression.			**5. FUNDING NUMBERS**
6. AUTHOR(S) Retsas, Ioannis			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Distribution Statement (mix case letters)			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

Digital watermarking is a highly evolving field, which involves the embedding of a certain kind of information under a digital object (image, video, audio) for the purpose of copyright protection. Both the image and the watermark are most frequently translated into a transform domain where the embedding takes place. The selection of both the transform domain and the particular algorithm that is used for the embedding of the watermark, depend heavily on the application. One of the most widely used transform domains for watermarking of still digital images is the Discrete Cosine Transform domain. The reason is that the Discrete Cosine Transform is a part of the JPEG standard, which in turn is widely used for storage of digital images. In our research we propose a unique method for DCT-based image watermarking. In an effort to achieve robustness to cropping and JPEG compression we have developed an algorithm for rating the 8×8 blocks of the image DCT coefficients taking into account their embedding capacity and their spatial location within the image. Our experiments show that the proposed scheme offers adequate transparency, and works exceptionally well against cropping while at the same time maintains sufficient robustness to JPEG compression.

14. SUBJECT TERMS Digital Image Watermarking, JPEG Compression, Discrete Cosine Transform			**15. NUMBER OF PAGES** 137
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A DCT-BASED WATERMARKING ALGORITHM ROBUST TO CROPPING AND COMPRESSION

Ioannis Retsas
Lieutenant, Hellenic Navy
B.S., Hellenic Naval Academy, 1991

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2002

Author: Ioannis Retsas

Approved by:
Ron Pieper
Co-advisor

Roberto Cristi
Co-advisor

David C. Jenn
Second Reader

Jeffrey B. Knorr
Chairman, Department of Electrical Engineering

Dan C. Boger
Chairman, Information Warfare Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Digital watermarking is a highly evolving field, which involves the embedding of a certain kind of information under a digital object (image, video, audio) for the purpose of copyright protection. Both the image and the watermark are most frequently translated into a transform domain where the embedding takes place. The selection of both the transform domain and the particular algorithm that is used for the embedding of the watermark, depend heavily on the application. One of the most widely used transform domains for watermarking of still digital images is the Discrete Cosine Transform domain. The reason is that the Discrete Cosine Transform is a part of the JPEG standard, which in turn is widely used for storage of digital images. In our research we propose a unique method for DCT-based image watermarking. In an effort to achieve robustness to cropping and JPEG compression we have developed an algorithm for rating the 8×8 blocks of the image DCT coefficients taking into account their embedding capacity and their spatial location within the image. Our experiments show that the proposed scheme offers adequate transparency, and works exceptionally well against cropping while at the same time maintains sufficient robustness to JPEG compression.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to acknowledge

- Dr. Ron. J. Pieper and Dr. Roberto Cristi, for their expert knowledge and guidance
- Evie, for her love and support and also for her priceless gift, our son Athanasios.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Watermarking is a method of providing protection of intellectual property in digital multimedia, and is based on hiding a digital signature within the data. With this signature one can identify the proprietor of a certain set of data and thus protect her/his intellectual property. In order for the watermarking to be dependable it is imperative that it has certain characteristics. The most important of these are: transparency of the watermark (it should be imperceptible to the Human Visual System), and robustness against common tampering with the image. This tampering may include JPEG compression, or cropping. With our work we provide a technique that gives satisfactory results in terms of transparency and robustness against JPEG compression and cropping. The new feature in our work has to do with the method we use for the matching of the image blocks and the watermark coefficients that are embedded in each block.

The embedding takes place in the DCT domain, which is also used by the JPEG standard, and allows for the exploitation of the domain's particular characteristics and the achievement of watermark transparency. Both the watermark and the image are DCT transformed. We have developed a method for rating the 8x8 blocks of the DCT of the image according to their Priority Coefficient (PC), which is a measure of their embedding capacity and their resistance to cropping.

For each 8x8 block of the DCT coefficients of the image, we calculate the Complexity Factor (CF), a novel metric for measuring the capacity of each block to receive watermark coefficients. We know that in the areas of the image where we have more “action” we can embed more information imperceptibly. In the literature there have been attempts to use the variance of each 8x8 block of the image as a measure of imperceptibility after watermark embedding. We show that the Complexity Factor as a capacity metric is a more accurate approach since the variance of the image blocks alone, does not necessarily manifest the actual visual properties of the particular spatial section of the image.

Additionally, for each block of the cover image we calculate the Center of Interest Proximity Factor (CIPF), which is a measure of significance of each 8x8 block with

respect to cropping resistance. We first determine the Euclidean distance *r*, between the center of the block, and the Center of Interest (CI). In our experiments we assumed that the CI is the center of the image. The Euclidean distance *r*, is then normalized over the diagonal (i.e. the maximum possible distance within the image) to produce a normalized value *rnorm*. This normalized distance is then processed by a transformer with characteristic function *f*,

$$f(rnorm) = -\frac{1}{\pi} \cdot \tan^{-1}(k \cdot (rnorm - \frac{2}{3})) + \frac{1}{2},$$

to result in the CIPF (CIPF=*f*(*rnorm*)). The distribution of the CIPF over the 8x8 blocks of a 256x256 image can be depicted in figure 1.

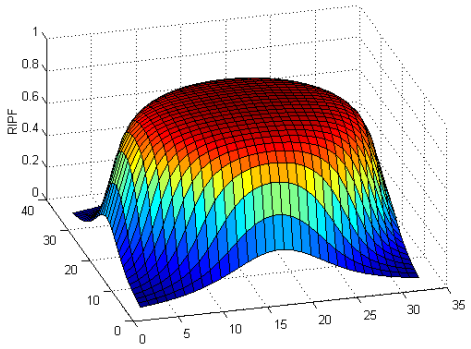

Figure 1. Distribution of the CIPF over the 8x8 blocks of a 256x256 image with k=15.

The CF of each 8x8 block is scaled by the CIPF to produce a Priority Coefficient (PC), which is attached to the block and contains all the information that is required for its rating. The blocks are now sorted by descending order of their PC.

The DCT coefficients of the watermark are sorted according to magnitude and divided into *m* groups of descending magnitude with equal number of elements. We then form embedding sets of coefficients. Each set contains *m* coefficients, one from each group. By this scheme, we

- embed the largest coefficients in the blocks with the larger capacity thus, ensuring transparency,
- avoid block saturation, which would very likely occur if only large coefficients were embedded in one block, and,
- protect the largest coefficients (which are the most important ones) by embedding them to the blocks which are more unlikely to be cropped.

The sets are then embedded into *m* frequency coefficients of the 8x8 DCT image blocks. Embedding in the lowest frequencies allows for higher robustness of the watermark against JPEG compression, since these coefficients are the least affected by the quantization process. However, the lower frequencies are the most perceptible ones, but we manage to compensate for the latter, by appropriately adjusting a weighting factor α.

The decoder works in reverse order and requires both the original image and the watermark. The level of detection is based on the correlation coefficient ρ, which is given by

$$\rho = \frac{\sum_i \sum_j W(i,j) Wr(i,j)}{\sqrt{\sum_i \sum_j [W(i,j)]^2 \cdot \sum_i \sum_j [Wr(i,j)]^2}},$$

and is a measure of similarity between the watermark W and the extracted pattern Wr.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

With the recent developments in digital communications and digital signal processing, and the expansion of the Internet, the proliferation of digital material (audio, images, video) has become extremely easy. The possible implications of this situation include the unauthorized distribution of such material with the purpose of making illegal profit or otherwise damaging the legal owner. Inevitably the business world and the authorities have expressed great concern over this issue, and as a result, the scientific community has become extremely active trying to provide techniques for copyright protection of digital material.

Towards this direction several types of secure communication methods are being explored addressing different aspects of the problem. These methods are either evolution of previously discovered techniques (types of encryption date back to the Roman era) or innovations that are dictated by the recent technologic developments.

Watermarking is a method of providing protection of intellectual property in digital multimedia, and is based in principle on hiding a digital signature (not to be confused with the term signature as used in cryptography) within the data. With this signature one can identify the proprietor of a certain set of data and thus protect her/his intellectual property.

A. PURPOSE

The purpose of this research is to investigate the application of the Discrete Cosine Transform in Digital Watermarking.

In this thesis we deal with watermarks for digital images. In order for the watermarking to be dependable it is imperative that it has certain characteristics. The most important of these are: imperceptibility of the watermark to human eye, and robustness against innocent or malicious tampering with the image. Among the most common ways of tampering with an image are: cropping, JPEG compression, resizing, filtering etc. It is these characteristics that dictate the continuous research on the field for the development of a robust scheme.

In general a watermarking technique involves the transformation of the image to a transform domain (FFT, DCT, DWT), if other than the space domain, and the embedding of the watermark coefficients on some or all of the image coefficients. The selection of the embedding domain has to do with the specific characteristics we want to exploit. The DCT domain in particular, is very popular in the watermarking community. The DCT is a part of the JPEG standard, and JPEG is in turn a very widely used image compression technique. By embedding the watermark in the DCT domain we can therefore create embedding schemes that are particularly robust against JPEG compression.

Through the course of this research a considerable amount of relevant work was examined and evaluated in terms of their results. Part of this work served as the basis for the development of our testing platforms. Starting from basic principles we have developed a complete watermarking scheme. Our scheme has been tested against different attacks and proved to be adequately transparent and robust. Additionally it has been tested for different embedding parameters and results have been produced and evaluated. Slight variations of the basic algorithm have also been developed and investigated in an effort to reach better results.

In this thesis we present a unique method for rating the 8×8 blocks of the image DCT coefficients according to their embedding capacity. Furthermore, an algorithm has been developed for determining the watermark coefficients that are embedded in each block. Our goal was to achieve maximum transparency and robustness against cropping and compression at the same time.

B. RESEARCH QUESTIONS

There are a number of research questions that we strive to answer in this thesis.

Firstly, we attempt to analyze the Discrete Cosine Transform and its potentials as a watermarking method. We investigate the watermark characteristics that affect the performance of a watermarking scheme and also the arguments for supporting perceptual or random watermarks.

Since in almost every watermarking transparency is paramount, we discuss the parameters that may be used for determining the capacity of each image block following ideas that have been suggested in the literature.

Identifying the factors that affect the quality of a watermarking scheme when under cropping or JPEG compression attacks was one of the basic elements of our research. The result was the development of the new algorithm that is proposed here. Consequently, the evaluation of the robustness of the proposed algorithm under attack became also one of the primary objectives.

C. THESIS OUTLINE

This thesis is organized as follows:

Chapter II provides the background required for the novice in the field. After a brief historic overview, Digital Watermarking is identified among other relevant technologies and the lines between these technologies are drawn. The needs that dictated the development of this technology are explained and also the requirements of a Digital Watermark that stem from these needs are reviewed. Definitions for terms and concepts pertaining specifically to Watermarking are given and they serve as a tool for better understanding the different approaches. Finally the different watermarking techniques that have been developed are reviewed with emphasis given in comprehending the principal differences between them. Brief examples of recent research work are given in order to support our arguments.

Chapter III involves a more technical insight of the technology, and the mathematical tools necessary for comprehension of our research are presented. In that context the Discrete Cosine Transform (DCT) is analyzed and its connection to the JPEG standard is discussed. The JPEG standard is reviewed and all is elements namely the DCT, quantization, and encoding are explained.

In Chapter IV the train of thought that led to the development of the new algorithm is shown. As our reasoning progresses, a step-by-step implementation of a new algorithm is revealed, and the way we attacked the problem is analyzed. This chapter is divided into three sections. In the first section new terms and concepts are introduced and explained. In the next section, we propose a new algorithm that deals with the transparency problem and offers sufficient robustness against cropping and JPEG compression. Finally we offer a description of the Watermark recovery process that was used in the proposed scheme.

Throughout our research several schemes were tried and evaluated. These schemes are presented in Chapter V regardless of their effectiveness because they can be the basis of future work. Finally the algorithm that was used in certain key elements of our scheme is analyzed.

Chapter VI presents experimental results validating the arguments in Chapter IV and V. We start from the images and the watermarks that were used, and the reasons why we chose these in particular. The results of our experiments are collectively presented here. We made an effort to present the results in such a way that they would better support the conclusions of the next chapter.

Finally in Chapter VII our work is briefly summarized and conclusions following the experimental results are made. Also we make suggestions for possible future work based on this material.

D. EXPECTED BENEFITS OF THE THESIS

Digital watermarking is a research area still being under exploration. None of the methods proposed so far has yet dominated, while the market is still in need of a dependable scheme that will provide watermarking robustness. It is yet not known if the development of a composite watermarking algorithm that will be used for different applications is feasible. So far it appears that even for objects of the same data type the watermarking algorithms that have been developed, seem to address very specific problems (for example in digital image watermarking the DCT based watermarks were primarily used to address the problem of JPEG compression). Towards this direction researchers all over strive to make all the necessary steps that will lead to a complete, dependable watermarking algorithm.

With our research we try to investigate how the different watermarking parameters affect the quality of our product. The issue of embedding the watermarks in selected image blocks, that allow imperceptible embedding is also addressed. Finally, we propose a new algorithm that may serve as the basis for further research in the field. We hope that this research contributes towards the direction of developing a composite image that addresses collectively all the possible attacks.

II. BACKGROUND ON DIGITAL WATERMARKING

A. HISTORIC REVIEW

The problem of achieving hidden communication between two parties has been investigated for thousands of years. One could safely assume that from the moment mankind formed organized military groups that were engaged in wars of any extent, the need for secure communications between members of the same group was probably experienced. There have been mainly two approaches towards a solution; cryptography and steganography. Both words are derived from Greek (cryptography: *κρυπτός* (=hidden) + *γράφειν* (=writing), steganography: *στεγανός* (=protected) + *γράφειν*). Their distinction is based on the following: cryptography is a way of communication, where the information to be secured is scrambled by the use of certain code, in a way that a third party, without the code, would be unable to retrieve the information; steganography on the other hand, is trying to achieve secure communication by hiding the existence of the message.

There is written evidence that secure communication techniques were exercised from as early as the years of Homer. The most frequently cited evidence though, is in the descriptions of the Greek historian Herodotus of Halicarnassus (440 BC). He states that a slave was sent by his master, Histiaeus, to Aristagoras the ruler of the city of Miletus. The slave was carrying a message for Aristagoras tattooed on his scalp. After tattooing the message he let his hair grow back again. Only when he had safely traveled to Miletus did the slave shave his head to reveal the message to Aristagoras encouraging him to revolt against the Persian King.

Aeneas the Tactician of Greece in one of his earliest books on military science, *On the Defense of Fortified places*, described as early as the fourth century, a system of cryptography. The Caesar Cipher attributed to the Roman emperor Julius Caesar (100BC – 44BC) was used for the communication between him and his generals. It was based on shifting each letter of the communicated text by a certain number of positions in the alphabet. The amount of shifting was known only to him and to his generals. For everybody else the message had absolutely no meaning. Petitcolas *et al.* in his work

Information Hiding – A Survey ([1]), does a considerable research on the use of secure communication techniques throughout history. Among others he mentions that the head shaving technique that was used by Histiaeus back in the classical Greece was also used by German spies in the beginning of the twentieth century.

As Ryan describes in [2] the Russian failure at Tannenberg in August of 1914 saw the complete destruction of two Russian armies by a single German army half their combined size. This decisive victory directly resulted from the fact that the Russian communications were compromised. The Russians had failed to distribute the military ciphers and their keys making it impossible for the two neighboring armies to securely communicate. All the Russian communications as the battle progressed were in the clear and therefore the Germans knew exactly what the Russian plans were, sometimes even before the Russian had received the orders by their command. The result, as Ryan clearly puts it, was that *in the end, 30,000 Russians were killed or missing, 100,000 were captured, one of the two Russian armies was devastated and one simply ceased to exist, all at the guns of the smaller but more mobile German army with its infinitely more secure communications.*

In the same work the author reveals that although the Japanese policy stressed the importance of communications security, their practices and procedures implementing that security were slipshod. Admiral Nimitz, thanks to the American cryptanalysis, was in hold of all the information that the captains of the Japanese ships knew about the battle of Midway. The advantage of surprise that Yamamoto depended upon was lost due to American cryptanalysts, and this cost the Japanese the battle, which turned the tide of the war.

Addressing specifically the watermarking history, we know that paper watermarks appeared in the art of hand papermaking nearly 700 years ago. According to Hartung and Kutter ([3]), the oldest watermarked paper found in archives dates back to 1292 and has its origin in Fabriano, Italy, which is considered the birthplace of watermarks. Thereafter, paper watermarking spread quickly all over Europe and beyond its use as a security feature, served also as an indication for paper format and quality. Paper watermarks were also used to date and authenticate paper.

Paper watermarking is still used, and is one of the major security measures in today's banknotes. The EURO (€) that was introduced only a few months ago among the European Union Countries was designed to have a watermark as one of its security features. In figure 2 we can see the watermark in the 5€ banknote.

Cox *et al.* in their recent book on Digital Watermarking ([4]) make a reference to the book "The Codebreakers", by Kahn, where there are stories of information hiding which are more relevant to watermarking. It is described in particular, that in the book "Hypnerotomachia Poliphili", which was anonymously published in 1499, there was a secret message hidden. Putting together the first letters of each chapter you would form the phrase "Poliam Frater Franciscus Columna Peramavit", which means "Father Francesco Columna loves Polia".

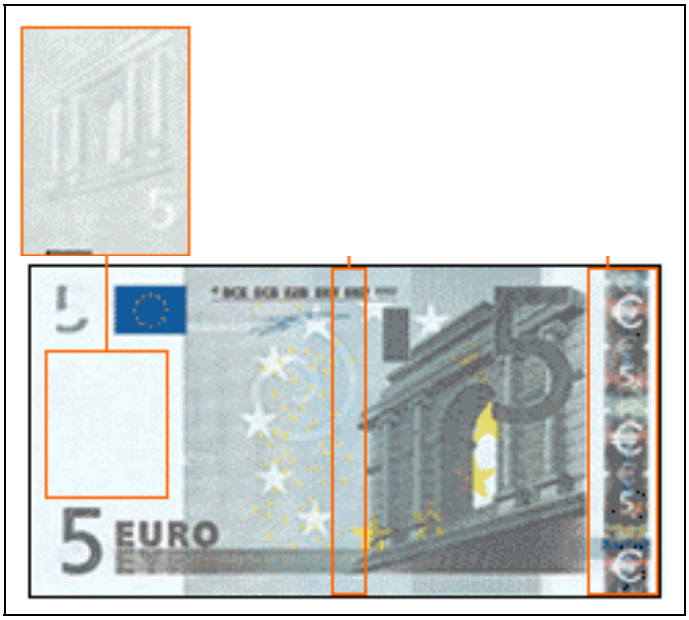

Figure 2. The 5 EURO banknote and its watermark (copied from the European Central Bank site for the new currency at http://www.euro.ecb.int/)

In the same book [4], the authors mention a story that takes place in the mid twentieth century and involves the use of a watermark very similarly to the way watermarks are used now, in the digital world. Specifically, in 1954, Emil Hembrook, of the Muzak Corporation, inserted an identification code in music by intermittently applying a notch filter centered at 1KHz. He used the Morse coding, and therefore by the

absence of energy that the filter caused, and by its duration, one could identify the hidden information. It is interesting to note that this invention is described by the US Patent in 1961, as an invention that makes possible the identification of the origin of a music presentation, therefore, preventing piracy.

There is clearly a connection between paper watermarks, steganography, and digital watermarking. In fact, paper watermarks in banknotes probably inspired the first use of the term watermarking in the context of digital data ([3]).

It is debatable who were the first to introduce the term *digital watermarking* ([4], [3]). What appears to be more accurate is the Cox version, which states that the first to use the term were Komatsu and Tominaga, in 1988. It took however a few more years until 1995/1996 before watermarking received remarkable attention. Since then, digital watermarking has evolved very quickly, something that can be verified by the amount of papers published on the subject.

Nowadays many corporations and institutions are active in the field. As an example we mention the International Standard's Organization (ISO) taking interest in the technology in the context of designing advance MPEG standard. The DVD and audio CD industries also strive to produce secure watermarks. The significance of the research going on can be perceived by the example of the SDMI foundation and Verance Corporation that threatened to bring a lawsuit against a scientific team that participated in a "public challenge", broke their algorithm, and attempted to publish the results in a paper titled "Reading Between the Lines: Lessons from the SDMI Challenge" by Craver *et al*.

B. GENERAL CONTEXT OF INFORMATION HIDING

In the literature there have been several attempts for categorizing the different methods of secure communications. These attempts do not always agree and in some cases may even be conflicting. The terms used are general by nature, and thus they are frequently overlapping. We will try to describe the idea behind these terms and give an overview of how they are related to each other, but we will avoid any attempt to form a strictly defined Secure Communications' tree.

Encryption was up to a few years ago the only available means for protection. It involves the scrambling of the data with a key, which makes it difficult (depending on the

quality and the complexity of the encryption algorithm) for an eavesdropper to gain any information on the content of the data that is being exchanged. A key (the same or a different one depending on the encryption algorithm) should also be used by the authorized recipient in order for the message to be decrypted. The message on the communication channel is, in this case, meaningless for those not in hold of the Key. An unauthorized third party, knowing that an encrypted message is being exchanged, should try to break the encryption algorithm. Depending on the type of the message sent, one can select the strength of the encryption algorithm. For example if the message is "attack the enemy on January 1st, at 01:00" an algorithm that will not be broken by January 1st will suffice. Beyond that point the attack has already commenced, and it is reasonable to assume, that the enemy has already found out. So the type of the message determines the cost one has to pay, in terms of time and resources, in creating an algorithm with the appropriate strength.

With this scheme we achieve protection during the transmission of the message but we have no protection whatsoever in a case where the contents of the message are publicly available but their redistribution is not authorized. An audio CD for example contains information that is readily available for public use but unauthorized copying of its contents is illegal.

1. Information Hiding

A widespread term describing a broad area of secure communication methods is *Information Hiding*. It is a general term that encompasses different kind of problems. All of these problems have as a common denominator the effort to prohibit an unauthorized third party from obtaining access to a message. Despite our feeling that Information Hiding should be used interchangeably with Communications Security this is not generally the case. In most of the literature Information Hiding is treated as a subcategory of Communications Security, along with Cryptography.

Information Hiding may refer to either making the information imperceptible or keeping the existence of the information secret [4]. Petitcolas *et al.* ([1]) describe Information Hiding as *traffic security*, treating it separately from encryption. According to the same authors this discipline also includes such technologies as: spread spectrum

radio communications (widely used in the military in an effort to keep secret the exchange of radio transmissions), temporary mobile subscriber identifiers (provide to some extend location privacy to users of digital telephones), and anonymous remailers (which conceal the identity of the sender of an e-mail message).

2. Steganography

Steganography is the art of trying to keep concealed the very existence of the communication channel. Some examples of steganographic attempts throughout history have been shown in Section A above. In general, steganography falls under the Information Hiding root. In the Information Hiding tree provided in [1], there are further subdivisions of steganography (Linguistic, Technical) that we are not going to cover.

3. Covert Channels

Covert channels are described as channels that were not designed for the purpose of exchanging of information. The term is primarily used in computer security and describes the method that is used by programs that communicate information to unauthorized parties.

The most common way to implement this idea, is by inserting a *Trojan horse* into a service program. The user is normally unsuspecting of the situation and when using the service program he automatically leaks information. To better appreciate this technique we will provide a particular example of covert channels: the *storage channels* ([5]). In multiuser systems, the operating system does not normally allow users to write to the same file at the same time in order to prevent its possible corruption. Every file in use is "locked", and thus any "write" request from other programs is rejected by the operating system. A covert channel can signal a 1-bit information by whether or not a file is locked. At this time the service program may be reading confidential data and the Trojan horse signals the data one bit at a time by blocking or not an irrelevant predetermined file. The only extra requirement for the implementation of this technique is that the service program (with the Trojan horse) and the unauthorized third party have a common timing source.

C. WATERMARKING

1. Watermarking in the Digital World

The developments in the networking technology and the worldwide web have significantly increased the risk of piracy. The situation now has very much evolved from the days were the only storage means was a tape, and any kind of reproduction resulted in copies that were degraded versions of the original object. Nowadays the multimedia data are available on the Internet in digital form, which allows for the reproduction of exact copies of the original. Additionally, copying devices are quite efficient, and most importantly, inexpensive, and therefore virtually anyone could afford its use. Considering both these factors we have all the necessary requirements to managing illegal distribution of digital multimedia and thus financially damage the legal owner of an object.

As already explained in the introduction, cryptography is not an adequate method when it comes to the protection of material that is publicly available but its redistribution is unauthorized. The watermarking technology potentially offers the solution to this problem.

Steganography and watermarking have been developed based on the same theoretical roots, that we want to keep a secret message hidden from an intruder. However there are conceptual differences between the two. Firstly, the latter requires extra robustness against attacks since our priority is to maintain the integrity of the secret message / watermark. In steganography, on the other hand, the assumption is that there will be no such attacks against the hidden message only because its very existence is secret. Any kind of attack on the object carrying the message is of no importance because it only serves as a cover of our real intentions. Here only the secrecy of the communication path is paramount. In watermarking, there are cases where we select to reveal the existence of a watermark on our object (the intruder knows that there is a secret message but does not know how to remove it), challenging, in a way, potential attackers. However, this may serve as a deterrence, since an intruder might not select to attack an image knowingly marked. In other words in steganography an intruder strives to detect the existence of a secret communication path and to retrieve the hidden information regardless of the effect on the cover object, while in watermarking an intruder aims at removing the watermark while at the same time maintaining the quality of the object.

We realize that the developments in the watermarking technology were dictated by the need for copyright protection of digital material. And this need resulted from the developments of other technological fields i.e. networking, storage and reproduction of digital data etc.

2. Requirements

There are many different everyday situations where the watermarking technology would be applicable. The first that comes to mind is proof of ownership. When you publish an image in the web and you want to retain the copyrights, you need to have a means of proving your ownership in a dispute. Registering the image with the Office of Copyrights and Patents would be the most appropriate action. However this is not always what people do, either to avoid the cost involved or simply to avoid extra paperwork. In general you want to put a digital signature in the object you own, in a way that only you can extract it. Any copy of the object would carry that same signature. If an adversary wanted to steal your property he would have to extract your watermark from the object, and maybe insert his own instead. This situation dictates one of the properties that this specific type of watermark should have: *robustness* against any kind of tampering with the image. The appropriate watermark should not be easily extracted from the image and if it did, the image should be so much degraded (in terms of quality) that would not serve any purpose to the unauthorized user.

Next, we describe a situation where the owner of an object makes a legal agreement for supplying his object to clients. We need a watermark to identify which of the clients broke this agreement and supplied the object to third parties. The watermark in this case serves as a serial number, it should be robust against attacks, and, at the same time, unique for each customer.

A different type of watermark should be used for verifying that a certain copy of an object is indeed a credible copy and that it has not been tampered with, in a manner that "critically" alters its contents. There are tamper-proofing techniques that accurately detect that an object has been tampered with [6]. But these techniques produce *"yes-or-no results to the question of tampering"* ([7]) and therefore they are not useful in all possible cases. The big question here is what kinds of changes need to be detected and what

changes are of no interest to us. A considerable amount of research has been conducted in the area, in an effort to produce a watermark, which would be *fragile* to certain types of modifications and resistant to others. This would allow the detection of particular types of tampering, for example the use of Adobe Photoshop to add a non-existing object to the image, while others such as JPEG compression should go undetected.

Other qualities generally required from a watermark stem from the type of the application. For example in digital sound or images it would be preferable for the watermark to be imperceptible to the human senses (ear or eye) so that the quality of the marked image is not compromised.

3. Terminology

a. Public and Private Watermarking

In the literature there have been several approaches to this issue. Petitcolas *et al.* in [1] define as *private* watermarking systems those that require at least the original image for decoding. The authors of the paper further define the *Type I* and *Type II* private watermarking systems. As Type I they characterize those systems that base their detection process on the possibly marked image and an exact copy of the original one. The Type II systems on the other hand, require also the watermark for the decoder. One more category, the *Semiprivate* watermarks, is also mentioned. Public marking requires neither the original image nor the watermark. Consequently it is a more challenging scheme but the decoder results are expected to be poorer because of the small amount of information that is available throughout the process.

Cox *et al.* in [4] seem to put the two terms in a more general, though also more complicated, perspective. According to them, in both cases the world can be divided into a group of *trusted individuals*, and the *public*, who are assumed to be potential adversaries. In private watermarking the public has no access to the watermarking data whatsoever. In public watermarking however, the public is only allowed to detect the watermarks. The way the terms are used here, refer to the security requirements of the application. Similarly the same terms can be used to describe watermarking algorithms and as such they describe algorithms that fulfill the

corresponding security requirements. The authors admit that in that sense the usage of the terms *public* and *private* is ambiguous.

We will follow the definition given by Petitcolas ([1]), which is accepted in most publications. In many publications the term *public* is used interchangeably with *blind*.

Blind watermarking appears to have more applications and to be much easier to use. You need to pass only the tested image through the decoder. Private watermarking algorithms, since they need at least the original image, they have to be supported by higher security requirements. On the other hand, the development of a private watermarking algorithm should also be generally simpler and the results are expected to be significantly better.

The blind watermarking techniques developed so far do not seem to be adequately effective, but both subjects are currently under research and we should expect better results in the near future.

b. Robust and Fragile Watermarks

The term *robust* watermark describes those watermarks remain detectable within an object in spite of significant levels of tampering of all kinds. The detection of the watermark comes down to the determination of the probability that the watermark is present in the object. In other words this is a measure of how confident we are that the tested object is indeed marked. Even when the detector gives a yes-no answer, in the general case, this results from the comparison of the calculated probability with a predetermined threshold. However, when an object is tampered with, it is automatically modified from the original, and in that sense its quality is degraded. Whether this degradation can be detected or not by the human sensors, is the question that needs to be asked. Therefore, we can define some limits for the maximum required robustness of the embedded watermark. The limits are set to the point where the object is subjected to so much tampering for the removal of the mark that the results not only can be detected by the human sensors, but also its quality becomes very low to offer any benefit to an attacker. In reality the situation is much more complicated because for each different kind of tampering the limits described above are different. To exhaust all possible attacks

(different kinds of tampering) and thus set a final limit that would cover all the cases if not impossible, it is not an easy task.

A *fragile* watermark has the purpose to confirm that the object has not been tampered with. In cryptography the same problem has been studied extensively and the most well known solution is the creation of a *digital signature*. In that sense the digital object is processed through a *Hash Function* [5]. A Hash Function "*produces a reduced form of the body of data such that most changes to the data will also change the reduced form"*. In particular a cryptographic hash function, uses a cryptographic function as part of the hash function. The sender in this case would evaluate the hash function of the data and send both the data and the hash value through the communication channel. A legitimate recipient should be in hold of the cryptographic algorithm. He should decrypt both the data and the hash value and then pass the data through the same hash function. By comparing the computed hash value with the value that was transmitted to him by the sender, he can verify that the data were received as sent. An intruder may be able to modify the data, or the hash, on their way over the channel. However, since he has only access to encrypted information, it is unlikely that he could modify both in such a way that they would match again.

As explained in one of the examples given in sub-section C.2 a watermark that potentially exhibits selective robustness, generally called *fragile* watermark, is required for tamper-proofing purposes. Again, the development of this kind of watermark faces serious problems. Except from the pure implementation issues that include the several different cases that need to be examined, there is also need for some limits to be set. These will define the cases where the watermark should be robust and the cases where it should be fragile. The lines are also in this case unclear and therefore difficult to be firmly established.

c. Fingerprinting

The term describes the watermarks that are used as a serial number on the copies of an object. They are like a fingerprint of the copy. The situations that dictate the need for their development are also described in sub-section C.2. The primary qualities of the fingerprints are robustness against attacks and uniqueness for each different copy of

the same object. According to [4], fingerprinting refers sometimes to the practice of extracting inherent feature vectors that uniquely identify the content.

D. IMAGE WATERMARKING TECHNIQUES

The driving force for the booming of the watermarking research was, as already explained, partially the Internet users, who are in need to secure their multimedia products that are available on the internet, and also the industry of musical CDs and DVDs that are even more desperate to protect their intellectual property and secure their profits. There is demand for all kinds of watermarks. In our research we will deal with digital images. The amount of research in this field (image watermarking) is larger compared to other fields and this is partially due to the large amount of digital images that are available in the Internet. There are two main embedding techniques different in principle: one that involves embedding in the space domain, and the other that uses instead a transform domain. In [3], [8], and [9] the authors provide an overview of some of the significant work in digital watermarking involving different embedding approaches.

1. Space Domain Watermarking

The space domain techniques are generally considered more susceptible to the various kinds of attacks. However these techniques were implemented first, and there is still research going on in the area, though not as intense as in the transform domains. Space domain techniques can be chosen for low cost schemes requiring low complexity and small computational overhead.

The early space domain watermarking techniques were not particularly efficient. One of the most primitive ideas was embedding in the Least Significant Bit of the pixel values. This technique is generally easy to detect and thus not much sophistication is required to remove the watermark. The space domain approach has evolved thereafter and methods have been proposed that are considerably more effective.

In [10] the authors propose the "Patchwork" method and the "Texture Block Coding". In the former randomly selected pairs of pixels (α_i, β_i) are used to hide 1 bit of watermark. The value of α_i is increased by 1 and the value of β_i is decreased by 1. For this method to work, some statistical properties should be satisfied. The latter involves

copying one image texture block to another area in the image with similar texture. For the recovery of the watermark the autocorrelation function is computed. This method has proved to be sufficiently robust to several kinds of distortion.

In [11] another technique is proposed. The authors use a binary watermark with equal numbers of ones and zeros, which has the same size as the original image. In half of the image pixels a binary one is embedded by changing the pixel number by an integer value k, which is the same for all the pixels. Hypothesis testing is used for the watermark detection and the method seems to behave well in down-sampling followed by up-sampling, and JPEG compression with compression ratios up to 1:4.

A somewhat improved version of this idea is proposed in [12]. The image is divided into non-overlapping 8×8 blocks. The blocks where the mark will be embedded are selected pseudorandomly. To each selected block a pseudorandom binary 8×8 block with equal number of ones and zeros is assigned. To embed a bit 1 the pattern is added to the block and to embed a zero the same pattern is subtracted from the block. Then the difference between the mean value of the image pixels that correspond to a 0 in the pattern is subtracted from the mean value of the pixels that correspond to a 1. The same calculations are repeated for the JPEG compressed counterpart of the image. If a 1 is embedded the differences from both the original and the compressed version need to exceed a threshold *T*. If a 0 is embedded both differences have to be below 0. If this requirement is not met the pattern is iteratively added or subtracted until the condition is met. This method is particularly designed for JPEG compression and according to the results presented, it seems to provide sufficient robustness.

Kutter *et al.* ([13]) attempt to embed a watermark in the space domain using only the blue image component in an RGB colorspace, in order to maximize the watermarking strength while keeping the visual artifacts minimal.

2. Transform Domain Watermarking

There have been several attempts by the research community to investigate the watermarking performance in different transform domains. The basic benefit from a transform domain technique is that by choosing a framework that matches the current compression standards, the watermarking algorithm can be designed to avoid embedding

in the coefficients that are normally discarded or severely quantized during compression. In this way we can ensure robustness to this particular kind of compression ([14], [15]).

In [16], to start from a rather unusual approach, the authors use the Fresnel transform to provide the embedding domain. The advantage of this approach is that several embedding plains exist in the Fresnel domain according to the various distance parameters thus providing many embedding channels. This work seems to give good results against certain geometric transformations and filtering but there is no indication whatsoever of its performance against any type of compression. In addition, no follow-up work has been observed in the literature.

In reference [17] the authors propose embedding in the DFT domain. In particular they select to embed the watermark using the phase of the DFT since it appears to be more important than the amplitude of the DFT values for intelligibility of an image. In other references ([18], [19], [20], [21]) the amplitude of the DFT is also used.

In reference [22] the authors propose a spread spectrum embedding technique, which uses the DCT domain as the embedding domain. Its innovation was how communication concepts such as spread spectrum can be applied to watermarking, and that the watermark can be embedded in the perceptually significant portion of the image. In spread spectrum communications, one transmits a narrowband signal over a much larger bandwidth such that the signal energy present in any frequency is undetectable. Similarly here, the watermark is spread over many frequency bins so that the energy in every one bin is very small and therefore unnoticeable. This idea can be applied to different transform domains. When the DCT is used, the transform is performed on the whole image and the watermark is embedded in the lowest frequency coefficients (excluding the DC component). As influential as this work may be, it has not yet produced the breakthrough method that the watermarking community is expecting.

A different idea is presented by Podilchuk *et al.* in [23], [24]. There, the concept of the Just Noticeable Difference (JND) is used. The JND thresholds have been used successfully in audio compression and in [24] the authors were the first to introduce the same concept to digital watermarking. In essence, the JND threshold determines the maximum level of distortion that will be transparent to the human visual system (HVS).

According to the authors there are three different properties of the HVS that determine these thresholds and need to be taken into account when building a model: (a) Frequency sensitivity, which describes the human eye's sensitivity to frequency gratings at various frequencies, and provides a basic visual model that depends only on viewing conditions and is independent of the content of the image; (b) Luminance sensitivity, which is a non-linear function for the HVS, and measures the effect of the detectability threshold of noise on a constant background; and (c) Contrast masking, which refers to the detectability of one signal in the presence of another signal. An attempt to incorporate the JND models to the work that is presented in this paper will be left for future work. This concept is applied to both the DCT and the Wavelet domain.

The authors, Piva *et al,* have also worked on a DCT-based method that exploits the masking characteristics of the HVS [25]. The watermark used is a pseudorandom sequence of N real numbers with normal distribution and the method appears to be effective with respect to JPEG compression median filtering and some geometric distortions.

The Wavelet domain appears also to be an appealing embedding domain. One reason being that it is included in the JPEG 2000 standards. Therefore wavelet-besed watermarking methods can be applied to provide protection against JPEG 2000 compression. Also the wavelet domain can be used as a computationally efficient version of the frequency models for the HVS ([26]).

THIS PAGE INTENTIONALLY LEFT BLANK

III. DCT DOMAIN TECHNIQUES

A. THE DISCRETE COSINE TRANSFORM

The Discrete Cosine Transform is a key element for JPEG compression and as such the related theory is important for our research. The concept is well explained in reference [27]. The DCT is a linear transform and therefore we will briefly introduce the linear transforms first.

1. Linear Transforms

Generally in a linear transformation we derive a sequence $\{\psi_n\}$ from a sequence $\{\chi_n\}$ based on the equation

$$\psi_n = \sum_{i=0}^{M-1} \chi_i \cdot \alpha_{n,i} \,. \tag{3.1}$$

Equation 3.1 is referred to as the forward transform. The original sequence can be recovered from the inverse

$$\chi_n = \sum_{i=0}^{M-1} \psi_i \cdot \beta_{n,i} \,. \tag{3.2}$$

We can get the same results using a matrix representation

$$\psi = \mathrm{A} \cdot \chi \tag{3.3}$$

$$\chi = \mathrm{B} \cdot \psi \,, \tag{3.4}$$

where $\chi = [\chi_0, \chi_1, ..., \chi_{M-1}]$, $\psi = [\psi_0, \psi_1, ..., \psi_{M-1}]$, and, A, B are $M \times M$ matrices with $[\mathrm{A}]_{i,j} = \alpha_{i,j}$, $[\mathrm{B}]_{i,j} = \beta_{i,j}$. The forward and inverse transform matrices A and B, are inverse of each other, and therefore $\mathrm{A} \cdot \mathrm{B} = \mathrm{B} \cdot \mathrm{A} = I$, where I is the identity matrix. Expanding these equations in two dimensions we get the general forward linear transform for a block of size $M \times M$

$$\Psi_{k,l} = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \mathrm{X}_{i,j} \cdot \alpha_{i,j,k,l} \,. \tag{3.5}$$

A two-dimensional transform is called *separable* if it can be decomposed into a sequence of one-dimensional transforms. In the case of images this leads to a transform of the rows, followed by a transform of the columns, or vice versa. In the separable case equation 3.5 can also be represented as

$$\Psi_{k,l} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} \alpha_{k,i} \cdot \mathrm{X}_{i,j} \cdot \alpha_{j,l} \ . \tag{3.6}$$

For a matrix representation again we have

$$\Psi = \mathrm{A} \cdot \mathrm{X} \cdot \mathrm{A}^T \tag{3.7}$$

and the inverse

$$\mathrm{X} = \mathrm{B} \cdot \Psi \cdot \mathrm{B}^T . \tag{3.8}$$

A transform is called *orthonormal* if the inverse of the transform matrix is the same as its transpose

$$\mathrm{B} = \mathrm{A}^{-1} = \mathrm{A}^{*T} . \tag{3.9}$$

Orthonormal transforms are energy preserving or in other words the sum of the squares of the original and the transformed sequences are equal. The proof is in reference [27] for the case of one-dimensional transform:

$$\sum_{i=0}^{M-1} \psi_i^2 = \psi^{*T} \cdot \psi = \left(\mathrm{A}\chi\right)^{*T} \mathrm{A}\chi = \chi^{*T} \mathrm{A}^{*T} \mathrm{A}\chi \tag{3.10}$$

For an orthonormal transform with transformation matrix A, it is implied that

$$\mathrm{A}^{*T} \cdot \mathrm{A} = \mathrm{A}^{-1} \cdot \mathrm{A} = I \quad , \tag{3.11}$$

and therefore

$$\chi^{*T} \mathrm{A}^{*T} \mathrm{A}\chi = \chi^{*T} I\chi = \chi^{*T} \chi = \sum_{n=0}^{M-1} \chi_n^2 . \tag{3.12}$$

Combining equations 3.10 and 3.12 we get

$$\sum_{n=0}^{M-1}\psi_n^2 = \sum_{n=0}^{M-1}\chi_n^2 \ . \tag{3.13}$$

2. The Discrete Cosine Transform

Among the several transforms that have been used in digital watermarking we will introduce the Discrete Cosine Transform, which is the basis of our technique. One can find sufficient details in several references ([27], [28], [29]). We will try to encapsulate the necessary information here in order to make it easier for the reader to follow the development of our research. We start from the one-dimensional case.

a. One-dimensional DCT

Given an array V of M numbers $V=[v_0, v_1, ..., v_{M-1}]$, let us define the sequence $V' = [v_0, v_1, ..., v_{M-1}, v_{M-1}, ..., v_1, v_0]$ where V' can be written as

$$V'[k] = \begin{cases} V[k], & 0 \le k \le M-1 \\ V[2M-1-k], & M \le k \le 2M-1 \end{cases} \tag{3.14}$$

We take the 2*M*-point DFT of V'

$$T[u] = \sum_{k=0}^{2M-1} V'[k] e^{-j\pi ku/M} = \sum_{k=0}^{M-1} V[k] e^{-j\pi ku/M} + \sum_{k=M}^{2M-1} V[2M-1-k] e^{-j\pi ku/M}, \ 0 \le u \le 2M-1 \tag{3.15}$$

Now, if we substitute $l = 2M-1-k$ in equation 3.15 we get

$$T[u] = \sum_{k=0}^{M-1} V[k] e^{-j\pi ku/M} + \sum_{l=0}^{M-1} V[l] e^{-j\pi(2M-l-1)u/M} \tag{3.16}$$

which yields

$$T[u] = e^{j\pi u/2M} \sum_{k=0}^{M-1} V[k] \left(e^{-j\pi(2k+1)u/2M} + e^{j\pi(2k+1)u/2M} \right), \ 0 \le u \le 2M-1 \tag{3.17}$$

and therefore

$$T[u] = e^{j\pi u/2M} \sum_{k=0}^{M-1} V[k] \cdot 2\cos\left(\frac{j\pi(2k+1)u}{2M} \right) \tag{3.18}$$

As a consequence of this result, we define a new transform

$$DCT(V[k]) = C[u] = \begin{cases} \frac{1}{2}T[0], & u = 0 \\ e^{-j\pi u/2M} \cdot T[u], & 1 \le u \le M-1 \end{cases} \tag{3.19}$$

From the preceding analysis we realize that the DCT of a vector V is derived if we take its mirror image, concatenate the two sequences to obtain a $2M$-point sequence, and then take the first M points of the resulting $2M$-point DFT.

The DCT pair is more commonly expressed as

$$C[0] = \sqrt{\frac{1}{M}} \sum_{k=0}^{M-1} V[k], \tag{3.20}$$

$$C[u] = \sqrt{\frac{2}{M}} \sum_{k=0}^{M-1} V[k] \cos\left(\frac{(2k+1)u\pi}{2M}\right), \tag{3.21}$$

for the forward transform, and for the inverse it can be shown ([28]) that

$$V[k] = \sqrt{\frac{1}{M}} C[0] + \sqrt{\frac{2}{M}} \sum_{u=1}^{M-1} C[u] \cos\left(\frac{(2k+1)u\pi}{2M}\right). \tag{3.22}$$

The variable in the argument of the cosine is responsible for frequency adjustments, and the factor that multiplies the cosine, adjusts the amplitude of the function. Clearly the IDCT is the summation of cosines of different frequencies and the DCT coefficients represent the amplitude of each cosine function.

b. Two-dimensional DCT

The two-dimensional DCT is defined as separable transform:

$$T[i,j] = c(i,j) \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} V[y,x] \cos\left(\frac{(2y+1)i\pi}{2M}\right) \cos\left(\frac{(2x+1)j\pi}{2M}\right), \tag{3.23}$$

where

$$c(i,j) = \begin{cases} \frac{1}{M}, & i,j = 0 \\ \frac{2}{M}, & \text{otherwise.} \end{cases}$$

The IDCT for the two-dimensional case is

$$V[y,x] = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} c(i,j)T[i,j]\cos\left(\frac{(2y+1)i\pi}{2M}\right)\cos\left(\frac{(2x+1)j\pi}{2M}\right). \qquad (3.24)$$

Unlike the DFT, the DCT is real and it is well known that compared to the DFT it is substantially better in energy compaction for most correlated sources. With the DCT we avoid the large coefficients for the high frequency components that are produced in the DFT due to the discontinuities at the boundaries.

B. THE JOINT PHOTOGRAPHIC EXPERTS GROUP (JPEG) STANDARD

The JPEG standard is one of the most widely used standards for lossy image compression and it offers a very good data compression rate. The standard proposed by the Joint Photographic Expert Group is based on the two-dimensional DCT and its components can be depicted in figure 3. The JPEG standard defines three lossy compression modes, namely, the *baseline sequential* mode, the *progressive* mode, and the *hierarchical* mode. The main difference between these modes is the way in which the DCT coefficients are transmitted. The *baseline sequential* mode, also called *baseline* mode for short, is the simplest of the modes and is required to be present in any case (even if other modes are used the *baseline* mode provides the default decoding capability [30]).

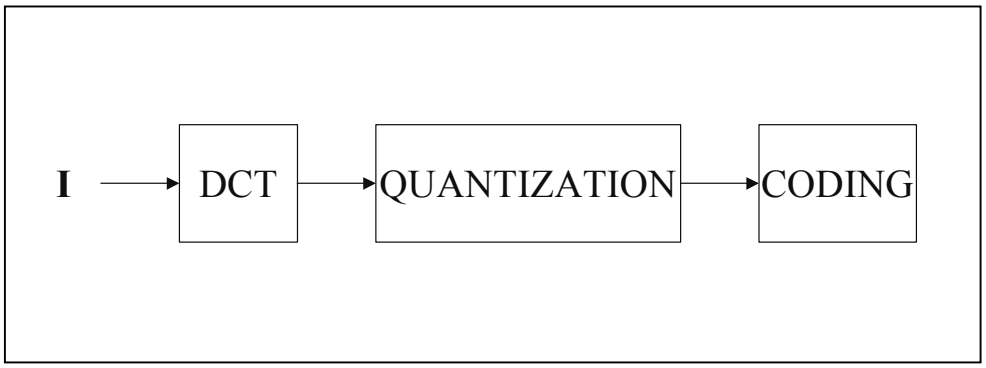

Figure 3. Block diagram of the JPEG compression.

1. The Transform

The transform used in the JPEG standard is the DCT transform described earlier. As a first step the value 2^{P-1} is subtracted from each pixel value, where P is the bit allocation per pixel. For the case of 8-bit per pixel images, the pixel values range from 0 to 255 and $2^{P-1} = 128$. This means that after the subtraction the pixel values are in the range [-128, 127]. This level shifting reduces the DC offset of the transformation (i.e. the value of the DC coefficient) but has no other effect whatsoever in the results. The JPEG standard dictates that the image is divided into non-overlapping 8×8 blocks and each block is then DCT transformed. In case that the image's rows or columns are not multiples of eight, the last row or column is replicated until the image reaches a multiple of eight size. Any additional rows or columns are discarded after decoding.

A more convenient method of expressing the DCT is in the form of matrix operations. In this case the forward DCT transform is

$$T = LVL^T, \tag{3.25}$$

and the inverse DCT is

$$V = L^T TL, \tag{3.26}$$

where L is given by equation 3.27.

$$L[i,j] = \begin{cases} \sqrt{\dfrac{1}{M}}, & i = 0,\ 0 \le j \le M-1 \\ \sqrt{\dfrac{2}{M}} \cos\left(\dfrac{(2j+1)i\pi}{2M} \right), & 1 \le i \le M-1,\ 0 \le j \le M-1 \end{cases} \tag{3.27}$$

This last form is particularly helpful for the implementation of the algorithm in computer programs and will be used indeed in our development.

2. Quantization

The next step after the DCT transform is quantization. In any case quantization is a lossy process and introduces distortion to the signal. It is obviously in our interest to maintain the distortion to a minimum.

The distortion introduced by quantization is measured by a distance metric. The most widely used is the Mean Square Error (MSE):

$$D = E[(x - \hat{x})^2] = \frac{1}{MN} \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} (x_{ij} - \hat{x}_{ij})^2 , \tag{3.28}$$

which applies to each $M \times N$ block. If we are interested in the size of the error relative to the signal, we can define the Signal-to-Noise ratio (SNR), as

$$SNR(dB) = 10 \log_{10} \frac{S^2}{D} , \tag{3.29}$$

where S is the average square value of the source output. A measure of the error relative to the peak value of the signal x_p is the Peak Signal-to-Noise ratio (PSNR), which is defined as

$$PSNR(dB) = 10 \log_{10} \frac{x_p^2}{D} . \tag{3.30}$$

The distortion introduced by quantization is inversely proportional to the step size, which in turn depends on the bits allocated per coefficient. Since the amount of information conveyed by each coefficient varies, it is reasonable to allocate different number of bits to each coefficient, with more bits to be allocated to the coefficients that carry more

information. As a measure of the amount of information that is carried by the DCT coefficients we can use the variance of the coefficients. Thus coefficients with larger variance are assigned more bits then coefficients with smaller variance.

The JPEG standard uses an 8×8 table called *quantization* table. The same quantization table is used for all the image blocks. The elements of the quantization table determine the step size that is used for the quantization of each coefficient in an 8×8 block. The JPEG standard allows different step sizes to be chosen for different coefficients. This implies that different amount of distortion is introduced for different frequencies. In general the higher frequency coefficients are more severely distorted with the use of greater step size. The decision of the relative size of the step size is based on repeated experiments that take into account the human psycho-visual system and the way the distortions in different frequencies are perceived by the human eye. In general errors in the higher frequency coefficients are more easily detectable and thus, in these frequencies we use larger step size.

The JPEG standard does not specify the exact quantization matrices that should be used, however, it suggests two quantization matrices, one for the luminance components, and one for the chrominance components that have proven to provide excellent results. One can create a customized quantization matrix that better suits one's needs. Tables 1 through 3 show examples of quantization matrices.

Table 1. The JPEG proposed luminance Q-table.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table 2. The JPEG proposed chrominance Q-table.

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

If $C[i,j]$ is the transform of an 8×8 luminance image block, its quantized counterpart $C_q[i,j]$ is given by

$$C_q[i,j] = \left\lfloor \frac{C[i,j]}{Q_L[i,j]} \right\rfloor, \tag{3.31}$$

where $Q_L[i,j]$ is the luminance quantization table, and $\lfloor \bullet \rfloor$ is the rounding division to the nearest integer. Then $C_q[i,j]$ is processed through a decoder to produce the reconstructed quantized coefficients $C_Q[i,j]$

$$C_Q[i,j] = C_q[i,j] \cdot Q_L[i,j]. \tag{3.32}$$

A quality factor q is normally used ([30]) to control the degree of quantization. This factor lies in the range [1, 100] and it represents the quality, expressed in percent, of the quantized image compared to the original one. A quantization factor c is then given by

$$c = \begin{cases} \dfrac{50}{q}, & 1 \le q \le 50 \\ 2 - \dfrac{2q}{100}, & 50 \le q \le 99 \end{cases} \tag{3.33}$$

Table 3. The luminance Q-table proposed by the IAHS Incorporation.

8	6	5	8	12	20	26	31
6	6	7	10	13	29	30	28
7	7	8	12	20	29	35	28
7	9	11	15	26	44	40	31
9	11	19	28	34	55	52	39
12	18	28	32	41	52	57	46
25	32	39	44	52	61	60	51
36	47	48	49	56	50	52	50

The standard JPEG quantization tables of Table 3.1 and 3.2 are used directly for q=50%. The same tables are multiplied by c to give the different quality (compression) levels. For 100% quality, q=100, that is lossless compression, and all the elements of $c \cdot Q_L$ are set to 1. A quantization example is given in figure 4.

Figure 4. *fishingboat* original (left) and quantized with quality factor 5% (right) (courtesy of the Signal and Image Processing Institute at the University of Southern California).

3. Coding

After the DCT transform and the quantization, further lossless compression is achieved by proper encoding. In each 8×8 block of quantized image coefficients the DC component is the top-left coefficient, while the highest frequency components are towards the bottom-right. In the general case after DCT the low frequency coefficients (top-left except the DC) have larger values as opposed to the low-frequency coefficients (bottom-right) that have smaller values. After quantization many of the coefficients towards the higher frequencies become zero. In order to group as many quantized zero-value coefficients together to produce longest runs of zero values, the AC coefficients are encoded using a zigzag path (figure 5). According to the JPEG standard the DC and AC coefficients are encoded separately.

a. DC Encoding

The DC coefficients tend to vary slightly between successive blocks. Therefore, only the difference, *DIFF*, between the current and the previous block is encoded. For the first block, the previous block value is set to zero. The potential value of *DIFF* varies in the range [-2040, 2040], however, in most cases *DIFF* takes relatively small values.

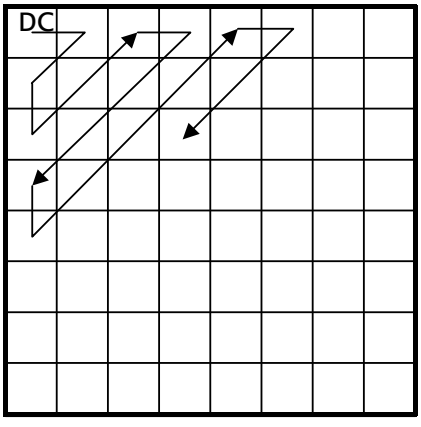

Figure 5. The zigzag path on an 8×8 block.

The number of bits *S*, that is required to represent *DIFF* is 1 to 11. Additionally although the difference of 0 requires 1 bit to express, it is represented as a special case with zero bits. Thus *S*=0 to 11 and can be broken into 12 categories. Table 4 shows the value of *S* for the different *DIFF* values. Now the compacted values are encoded with the use of Huffman code ([37]). The codeword consists of three parts: the code for *S* as obtained from Table 5; one sign bit, 1 for positive and 0 for negative; and the *S*-1 least significant bits of the *DIFF* value. If the *DIFF* value is negative, in the third part of the codeword we use instead the 1's complement of the *S*-1 least significant bits of the *DIFF* value. In the special case of *S*=0 the codeword consists of only one part, the Huffman code for *S* as obtained from Table 5.

Table 4. The *DIFF* categories.

S	Difference values
0	0
1	-1,1
2	-3,-2,2,3
3	-7,-6,-5,-4,4,5,6,7
4	-15,…,-8,8,…,15
5	-31,…,-16,16,…31
6	-63,…,-32,32,…,63
7	-127,…,-64,64,…,127
8	-255,…,-128,128,…,255
9	-511,…,-256,256,…,511
10	-1023,…,-512,512,…,1023
11	-2047,…,-1024,1024,…,2047

b. AC Encoding

For the encoding of the AC coefficients the zigzag path described earlier comes into use. The idea is that the quantization produces large blocks of successive zeros especially towards the high frequencies. Here we use a combination of Huffman coding and Run-Length coding. As we follow the zigzag path, each non-zero coefficient is described by a composite *R*/*S* symbol: *R* is a 4-bit element specifying the number of zeros between the last non-zero and this coefficient; and *S* is the number of bits that are required to express the non-zero coefficient as in Table 6.

If all remaining AC coefficients are zero the End-of-Block (EOB) symbol is set. If the number of zeros in a run exceeds 16 then the zero count recommences. Usually two hexadecimal symbols are used to represent the composite *R*/*S*. The codeword is again completed with two more parts; the 1-bit sign, and the *S*-1 last significant bits of the value. These are used in the same manner as in the DC case, which means that the third part is substituted by its 1's complement in the case of a minus sign.

Table 5. The Huffman code for *DIFF* values.

	DC Luminance		DC Chrominance	
S	Length	Codeword	Length	Codeword
0	2	00	2	00
1	3	010	2	01
2	3	011	2	10
3	3	100	3	110
4	3	101	4	1110
5	3	110	5	11110
6	4	1110	6	111110
7	5	11110	7	1111110
8	6	111110	8	11111110
9	7	1111110	9	111111110
10	8	11111110	10	1111111110

11	9	111111110	11	11111111110

Table 6. The AC categories.

S	Coefficient values
1	-1,1
2	-3,-2,2,3
3	-7,-6,-5,-4,4,5,6,7
4	-15,…,-8,8,…,15
5	-31,…,-16,16,…31
6	-63,…,-32,32,…,63
7	-127,…,-64,64,…,127
8	-255,…,-128,128,…,255
9	-511,…,-256,256,…,511
10	-1023,…,-512,512,…,1023

IV. A NON-UNIFORM WATERMARKING ALGORITHM

Through the course of our research we tried to reproduce some basic watermarking scheme that would adequately sustain the basic attacks of cropping and compression, while at the same time maintain sufficient transparency. Towards that end we formed some new theoretical concepts for the development of a new algorithm. Both the concepts and the algorithm are presented in this chapter.

A. ANALYSIS OF THE NEW CONCEPTS

1. Center of Interest Proximity Factor

We first processed the idea of rating the 8×8 blocks of the image DCT coefficients. The motivation for this approach was the insufficient performance against cropping that was evident in many of the studied schemes.

We assert that the resistance of the image to cropping depends heavily on the spatial location of the image blocks that are selected for embedding the watermark coefficients. If the coefficients are embedded on portions of the image that will be later cropped, those coefficients will be permanently lost. It is important to note that by transforming an image to the DCT domain in blocks of 8×8 pixels (JPEG standard), the spatial relation of each of the blocks of DCT coefficients is maintained.

It is generally correct that in cases of commercially used images there is a Region of Interest (RI), where most of the image information is concentrated. For the purpose of our analysis for each given image we determine a specific point, which is called the Center of Interest (CI). As the CI we may choose either the center of the image ($M/2,N/2$ for an $M\times N$ image), or any other point of the image. In the experiments to follow as the CI we used the center of the image. Following the same rationale it is reasonable to assume that anyone who would try to crop the image for any reason (either for attacking our watermarking system or just because he has no interest in the whole image), he would crop some portion near the borders of the image maintaining most of the information that is carried around the Center of Interest. Similarly, pie type cropping (figure 6) should probably be considered impracticable for anyone who would try to benefit from the

image. Our intention is to develop a method that takes into account the significance of the region of interest in determining the 8×8 blocks where the watermark will be embedded.

Figure 6. Peripheral (left) versus pie type (right) cropping of Lena (courtesy of the Signal and Image Processing Institute at the University of Southern California).

For each 8x8 block of the cover image we determine the Euclidean distance *r(m,n)* between the center of the block with coordinates (*m,n)*, and the CI (with coordinates (*M*/2,*N*/2) if the center of interest is the same as the center of the image). This distance is then normalized over the diagonal (i.e. the maximum possible distance within the image) to produce a normalized value *rnorm*, where *rnorm* $\in$ [0,1]. This normalized distance is then processed through a transformer with characteristic function *f*,

$$f(rnorm) = -\frac{1}{\pi} \cdot \tan^{-1}(k \cdot (rnorm - \frac{2}{3})) + \frac{1}{2}, \qquad (4.1)$$

where *k* can typically vary in the range [10,25]. The result is the Center of Interest Proximity Factor (CIPF=*f*(*rnorm*)). A typical distribution of the CIPF can be depicted in figure 7.

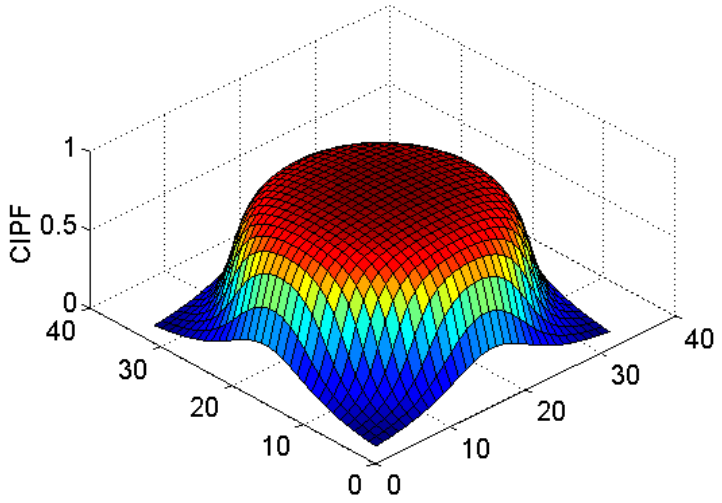

Figure 7. The distribution of the CIPF over the 8x8 blocks of a 256×256 image with k=15. The x and y axes are the coordinates of the image blocks (32×32 blocks in an 256×256 image).

2. Complexity Factor

The main idea is to embed the watermark in the image blocks where it could not be detected by the HVS. In the literature there have been several papers addressing this issue. One such attempt is to use the variance of the image blocks in the space domain as a measure of their embedding capacity ([31]). This means that if the variance of a block in the space domain is higher, we can embed in this block larger watermark coefficients, with lower probability that the produced distortion will be detected by the HVS. We claim that this is not quite correct, and we can prove the validity of our claims with a trivial example. In figure 8, both blocks (8×8) have the same number of black and white pixels. In spite of having different pixel arrangement, both blocks have the same variance (0.2540) in the space domain. We can obviously tell that changing any one pixel on the left block will be immediately detected by the human eye, whereas, the same alteration on the right block (which has a more complicated visual pattern) would require more thorough observation for detection.

Our idea is to weight the absolute value of the DCT coefficients of an image block differently, according to the part of the spectrum that they describe, and then add them up to produce a Complexity Factor (CF) for each block. With our method for the

same example of figure 8, we get a factor of 44.2044 for the left block against a factor 790.8275 for that on the right.

In our method we have excluded the DC coefficient from the calculations. The reason is that the DC coefficient represents the average pixel offset rather than frequency, and therefore should be ignored. Thus, we create the vector *weight*=[1,2,…,63], which we use to weigh the absolute value of the DCT coefficients using the formula

$$CF_i = weight \cdot \left| D_i' \right|, \tag{4.2}$$

where D_i is a vector (1×63) containing the DCT coefficients of the i^{th} block of the image according to the standard zigzag arrangement, (·) is the matrix multiplication operation, and CF_i is the resulting Complexity Factor for that block.

B. ENCODER

The encoder is described in principle by the block diagram in figure 9. Both the watermark and the image are DCT transformed and processed through the embedding algorithm. The outcome is then IDCT transformed and normalized to compensate for any errors that exceed the allowed limits of the pixel values

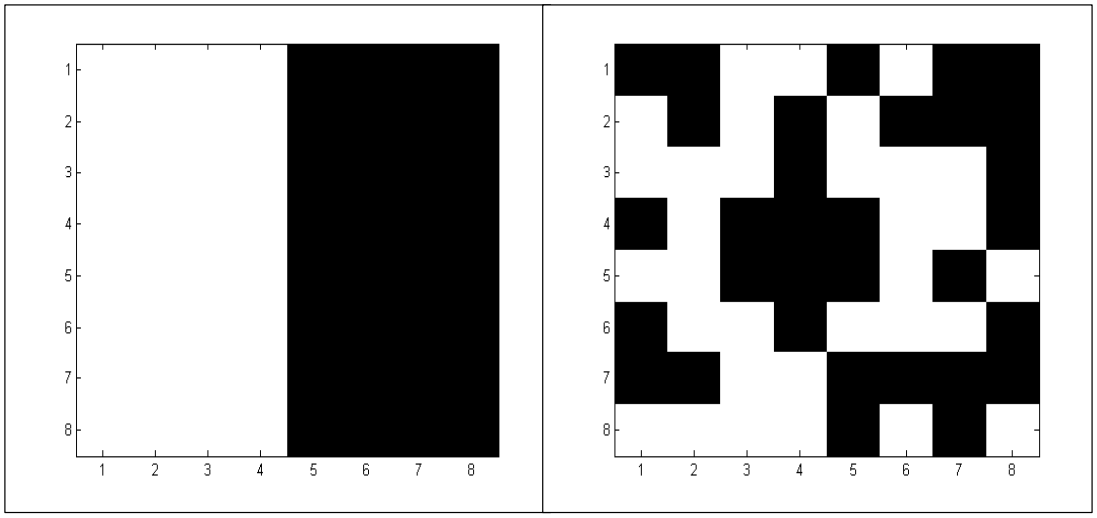

Figure 8. Two binary 8×8 blocks with the same number of ones and zeros but different perceptual characteristics.

1. Priority Coefficient

In Section A we saw how the *CIPF* and the *CF* are calculated for each image block. Now, we introduce a new coefficient, the Priority Coefficient (*PC*). Each image block *i*, is associated with a Priority Coefficient PC_i. The PC_i is defined as the $CIPF_i$, scaled by the CF_i.

$$PC_i = CIPF_i \cdot CF_i . \tag{4.3}$$

The image blocks are sorted according to descending order of their PC_i, to produce the sequence of blocks $B_1, B_2, \ldots, B_K$, (K is the total number of image blocks). Blocks that come first in the sequence are less likely to be cut off after cropping, and have larger variance in the lower and middle frequency coefficients allowing for higher unnoticeable distortion. Thus, they are capable of successfully "hiding" higher watermark coefficients or in other words they have larger embedding capacity.

2. Embedding Algorithm

In each block of the image DCT coefficients we embed a certain number of watermark DCT coefficients. In order to preserve transparency, we embed a small number of watermark coefficients in each 8×8 image block. We refer to the number of watermark coefficients that are embedded in one image block as the *embedding size* (*es*). A typical embedding size is 2, 4, or 8 watermark coefficients per image block. This means that the watermark size cannot be larger than $m \cdot K$, where m is the embedding size ($m \in [2,4,8]$), and K is the total number of 8×8 blocks in an image.

We tried to produce a scheme that embeds the watermark coefficients into the image blocks in the most efficient way. The rationale can be described by the following rules:

- *The watermark coefficients with higher magnitude should be embedded in the higher-rated image blocks*.

  This serves two purposes: higher magnitude watermark coefficients are in the general case the most important ones and as such they need higher protection against cropping; additionally, they cause greater distortion to an image block

after embedding, and as already explained the higher rated image blocks are less susceptible to distortion.

- *Not all the higher magnitude watermark coefficients should be embedded in one image block*.

  Otherwise the distortion in that block will be severe and will not be tolerated by the HVS.

Based on these rules we created the algorithm illustrated in figure 10. The DCT coefficients of the whole watermark are sorted according to descending order of magnitude [$c_1, c_2, ..., c_L$], where L is the total number of watermark coefficients. They are then divided into m groups with equal number of coefficients [$c_1, c_2, \ldots, c_{(L/m)} \mid c_{(L/m)+1}, \ldots, c_{(2L/m)} \mid \ldots \mid c_{((m-1)L/m)+1}, \ldots, c_L$]. The coefficients are now regrouped to form the embedding sets. Each set contains m coefficients, one from each group. The first coefficient from the first, second, … , m^{th} group form the first embedding set $es1$=[$c_1, c_{(L/m)+1}, \ldots, c_{((m-1)L/m)+1}$], the second coefficient from the first, second, … , m^{th} group form the second embedding set $es2$=[$c_2, c_{(L/m)+2}, \ldots, c_{((m-1)L/m)+2}$], and so on, until $es(L/m)$= [$c_{(L/m)}, c_{(2L/m)}, \ldots, c_L$]. The result is ($L/m$) sets sorted according to descending order of embedding weight from $es1$ to $es(L/m)$. The sets that come first in the list require image blocks with larger capacity. Therefore we embed $es1$ to $B1$, $es2$ to $B2$ etc.

Each set is embedded into m coefficients of the corresponding image block following the formula:

$$\left.\begin{aligned} u'_{i(xstart)} &= u_{i(xstart)} + \alpha \cdot C_{i1} \\ u'_{i(xstart+1)} &= u_{i(xstart+1)} + \alpha \cdot C_{i2} \\ u'_{i(xstart+m-1)} &= u_{i(xstart+m-1)} + \alpha \cdot C_{im} \end{aligned}\right\} \qquad (4.4)$$

where α is a weighting factor that typically ranges around 0.1, C_{ij} is the j^{th} coefficient of the i^{th} embedding set, u_{ij} is the j^{th} coefficient on the zigzag arrangement of the i^{th} block, and u'_{ij} is the modified image coefficient u_{ij} after embedding. For example, for m=4, c_5

corresponds to C_{21}. The index *xstart* denotes the value of j in u_{ij} where the embedding starts.

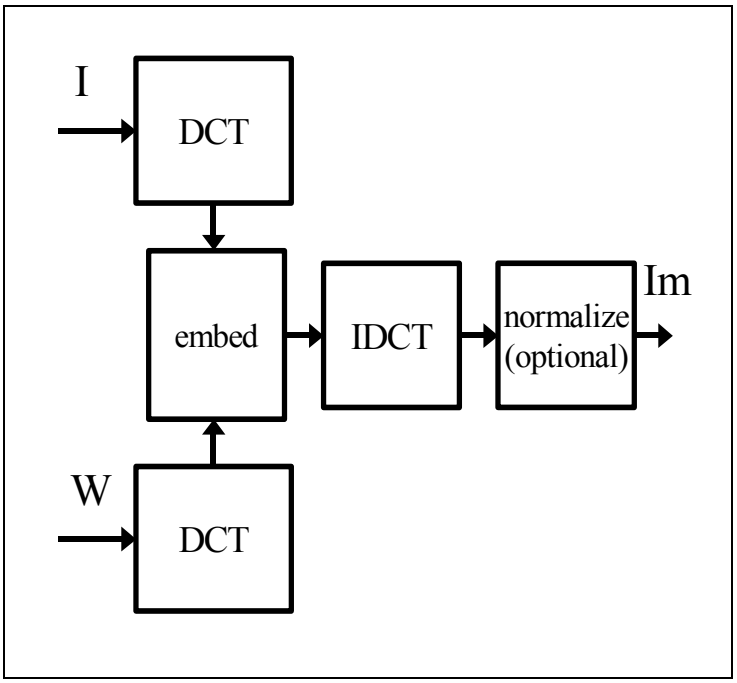

Figure 9. The basic encoder.

Normalization may be used as the last step that takes place in the space domain after the marked image coefficients have been IDCT transformed and have produced the marked image. The use of normalization is optional and the concept is further explained in the next chapter.

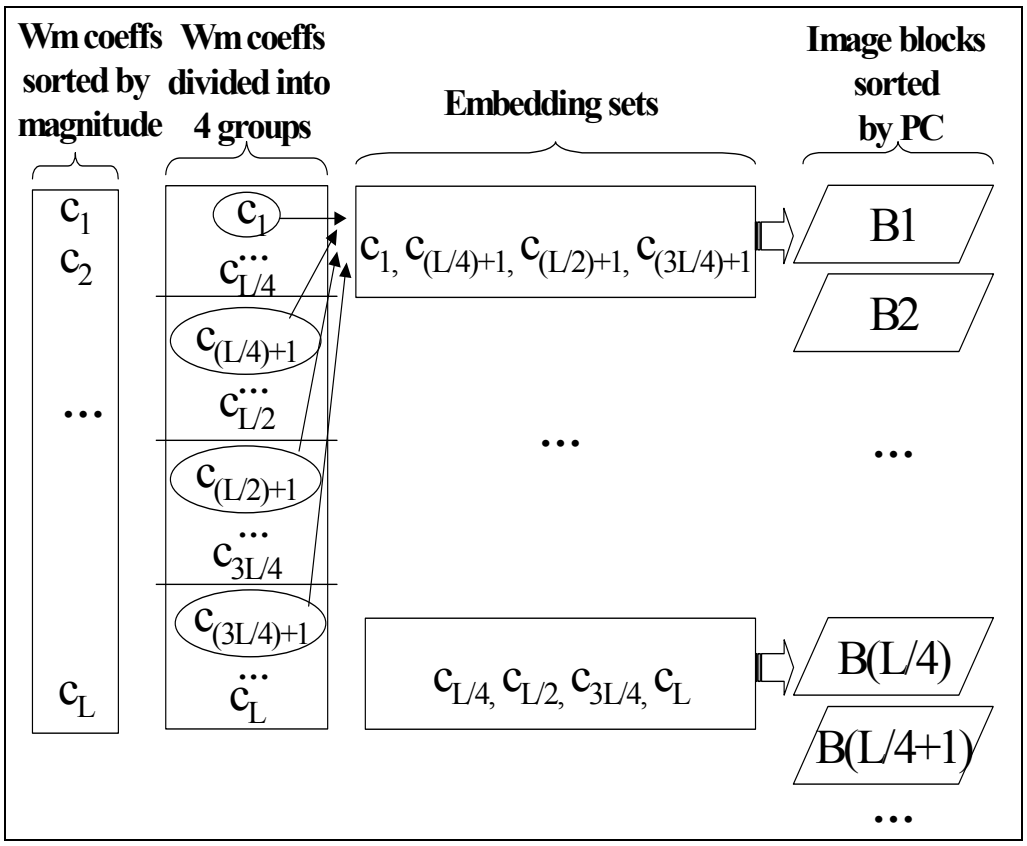

Figure 10. The algorithm applied for a watermark with L coefficients and embedding size 4.

C. DECODER AND DECISION MAKING

The decoder (figure 11) works in reverse order and requires both the original image and the watermark. The DCT coefficients of the test image are subtracted from the DCT coefficients of the original. At this stage the sorting information of the watermark coefficients is also needed to correctly reassemble the potentially recovered watermark. The result is IDCT transformed to produce the *recovered object*, *Wr* (product of the decoding process).

The decision making device is based on classical detection theory ([32]). The recovered object is now compared to the original watermark by calculation of the correlation coefficient ρ, of the two:

$$\rho = \frac{\sum_i \sum_j W(i,j) Wr(i,j)}{\sqrt{\sum_i \sum_j [W(i,j)]^2 \cdot \sum_i \sum_j [Wr(i,j)]^2}}, \tag{4.5}$$

where *W(i,j)* is the *(i,j)* pixel of the original watermark, and *Wr(i,j)* is the *(i,j)* pixel of the recovered object. The decoder decides whether the recovered object corresponds to an actual watermark or not, based on a predetermined threshold *T*. Higher ρ means that *W* and *Wr* are highly correlated and therefore have higher similarity to each other. This is interpreted as higher confidence that the processed image has been indeed watermarked. In the case where *W* and *Wr* are independent, ρ is normally distributed with zero mean ([33]). Therefore, the probability of ρ exceeding a certain threshold can be directly obtained from the normal distribution. The threshold can be accordingly adjusted to match our probability of detection, P_D, and probability of false alarm, P_{FA}, requirements.

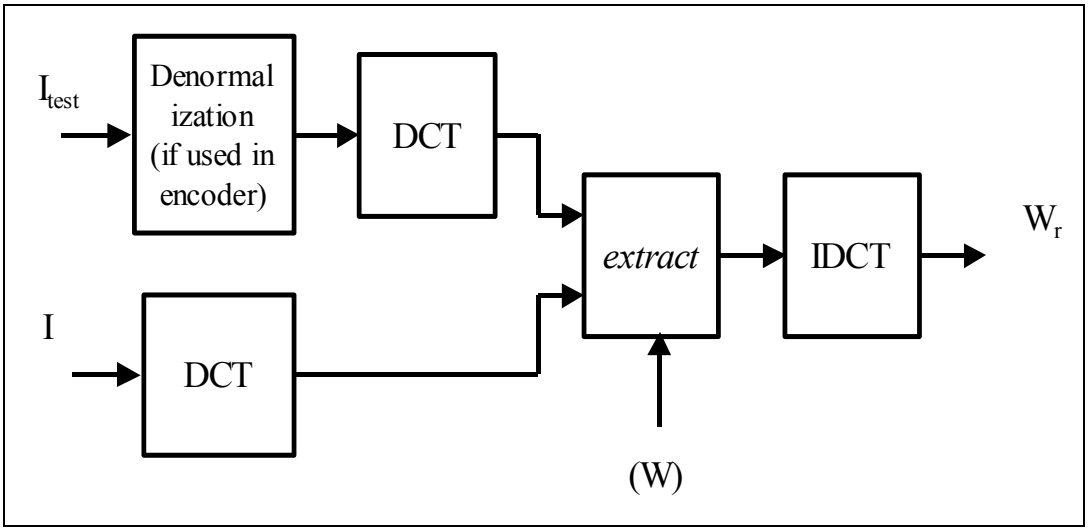

Figure 11. The decoder.

THIS PAGE INTENTIONALLY LEFT BLANK

V. IMPLEMENTATION ISSUES

In this chapter we describe several interesting parts of our research. Not all of them were successful, and not all of the successful ideas of this chapter were actually incorporated in the model presented in chapter four.

A. KEYING

To enhance the security of the watermark we can use a unique key. The key should be applied directly on the watermark, after the error correction code if any, but before any other processing. In the case of a random watermark however, the use of the key as a security feature is redundant or the key should also be the watermark itself. We can extend spread spectrum techniques ([4]) to watermarking by applying a key with length multiple of the watermark.

We implemented the keying feature in our algorithm using the following process: We started from an $M \times N$ grayscale watermark, W, with pixel values, $W[i, j]$ $(1 \le i \le M, 1 \le j \le N)$, integers in the range [0, 255]. Each pixel is translated into binary with 8 bits per pixel ($W_b[i, j, k], 1 \le i \le M, 1 \le j \le N, 1 \le k \le 8$). Essentially we now have an $M \times N \times 8$ binary matrix W_b. An equally sized binary Key, $K[i, j, k]$, is also produced, and the two are XORed

$$W_{bk}[i, j, k] = W_b[i, j, k] \oplus K[i, j, k], \quad 1 \le i \le M, 1 \le j \le N, 1 \le k \le 8. \tag{5.1}$$

The binary W_{bk} is translated back into 8-bit integer representation to produce the keyed watermark W_k. The concept is implemented in our algorithm with two functions; *keying* and *bitPlanes* (appendix A). The former produces the Key and performs the actual XOR operation while the latter decomposes the watermark into 8 binary planes (figure 12) with the plane at the back containing the Least Significant Bits (LSB) of each pixel value that has been translated into binary.

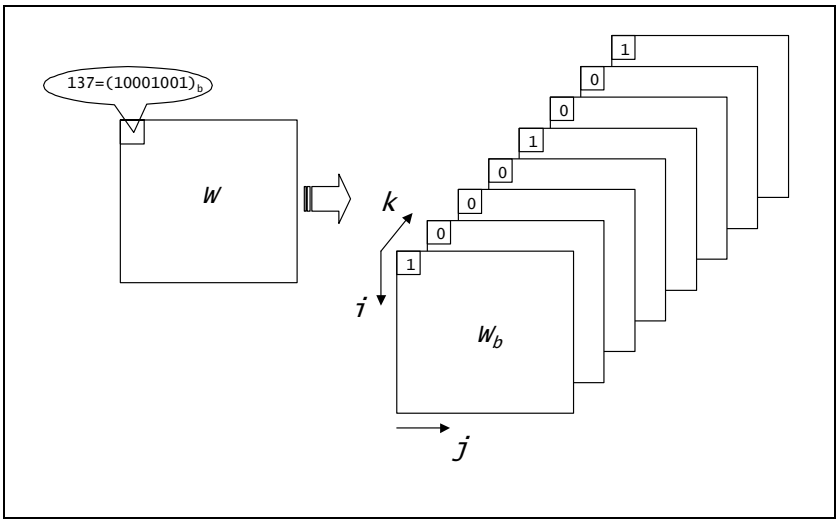

Figure 12. The *bitPlanes* function concept.

B. QUANTIZATION

We know from Chapter III that in JPEG compression the coding part is lossless and the errors that occur during the process are introduced by the quantization element (quantization noise). In order to test the performance of the proposed algorithm against JPEG compression it was therefore sufficient to reproduce the quantization component of the compression algorithm.

Since all our experiments were conducted with grayscale images we used the standard JPEG luminance quantization table. In essence equations 3.31 and 3.32 were modified to

$$C_q[i,j] = \left\lfloor \frac{C[i,j]}{cQ_L[i,j]} \right\rfloor, \tag{5.2}$$

$$C_Q[i,j] = C_q[i,j] \cdot cQ_L[i,j], \tag{5.3}$$

where c is given by equation 3.33 and Q_L is the standard JPEG luminance quantization table of Table 3.1.

C. NORMALIZATION

In the case where we have to work with parts of an image that are close to the limits of the pixels' dynamic range [0, 255], embedding the watermark coefficients and returning back to the space domain may produce results that exceed this dynamic range (figure 13 left). The most straightforward solution is to truncate all the off-range values to either 0 or 255. This approach may produce acceptable results in terms of transparency, but it introduces additional irreversible errors to the decoder, thus reducing the performance of the system.

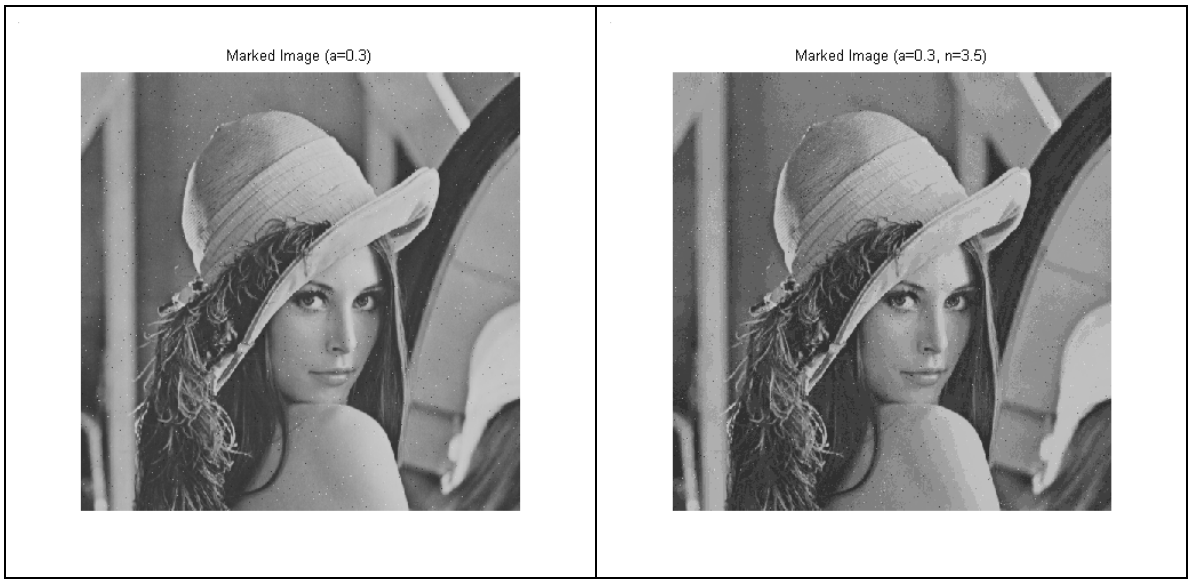

Figure 13. Lena marked (left) and marked and normalized (right). The black and white dots that can be seen in the left image are considerably fewer in the normalized image. In this case *stripes* was used and the watermark coefficients were randomly distributed throughout the image.

In cases where we are allowed to increase the bit allocation, we could use an invertible normalizer (figure 13). We devise a normalization function that maintains the dynamic range of the pixel values within the allowable limits. Our choice for the forward and the reverse function was

$$y = \frac{1}{\pi} \cdot \tan^{-1}(n \cdot (x - \frac{1}{2})) + \frac{1}{2}, \tag{5.2}$$

$$x = \frac{1}{2} + \frac{\tan(\pi \cdot (y - \frac{1}{2}))}{n}. \qquad (5.3)$$

The normalization parameter n is used to adjust the steepness of the curve. The value of 3.5 was experimentally proven to work better since it provides good pixel transformation (no shift) in the mid ranges. The characteristic curve of equation 5.2 for n=3.5 is shown in figure 14. As shown in the figure, the curve is limited within the range [0, 1]. This can be adjusted to [0, 255] by multiplication of equation 5.2 by 255. Appropriate modifications should also be made to the inverse function (equation 14).

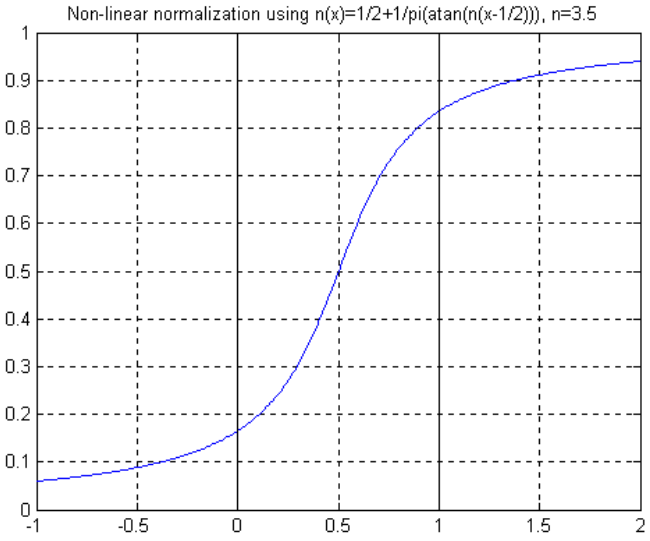

Figure 14. The normalization function for n=3.5.

The possible price to pay for the use of normalization is that the parts of the image with pixel values near the limits of the dynamic range are altered and their pixel values are shifted towards the center of the range. There are cases where this shift is perceptible to the HVS and therefore the success of the method depends heavily on the histogram of the image.

Any invertible function may be used instead, provided that it is limited within the allowable range. The effectiveness of the function is measured mainly by the quality of the normalized image or in other words by whether any changes are perceptible to the

HVS. We would like a good normalization function to be linear with a slope of 1 (θ=45°) for the most part and non-linear only at the boundaries.

In figures 15 and 16 we show the effect of the normalization on two images, namely *pentagon* and *arctic hare*, with intensity histogram shown. In figure 15 we see that the histogram of *pentagon* is concentrated towards the middle of the [0, 255] range. Our function in this case, works particularly well and the normalization in the right image is virtually undetected. The changes in the histogram, which indeed occur as we observe comparing the two plots, are imperceptible by the HVS.

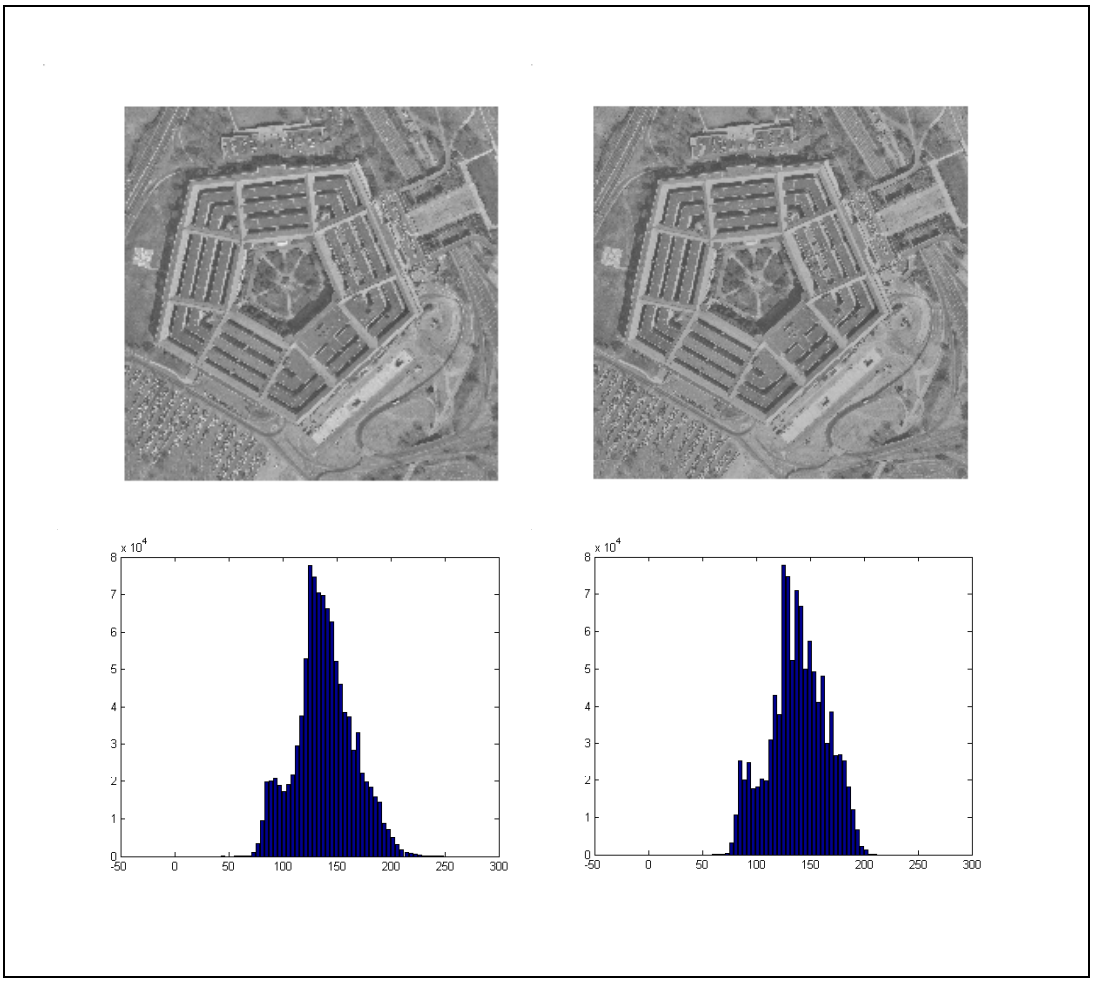

Figure 15. The original (left) and the normalized (right) *pentagon* and their corresponding histograms (courtesy of the Signal and Image Processing Institute at the University of Southern California).

In figure 16 we clearly see the effect of normalization in images with many pixels towards the boundaries of the allowable range. In *arctic hare* the white shades (pixel values near 255) are dominating the image and therefore the histogram displays a large concentration of pixels towards the right end. The characteristic function that we used, performs poorly in this extreme situation as it imposes significant changes to most of the pixel values after processing through the normalizer.

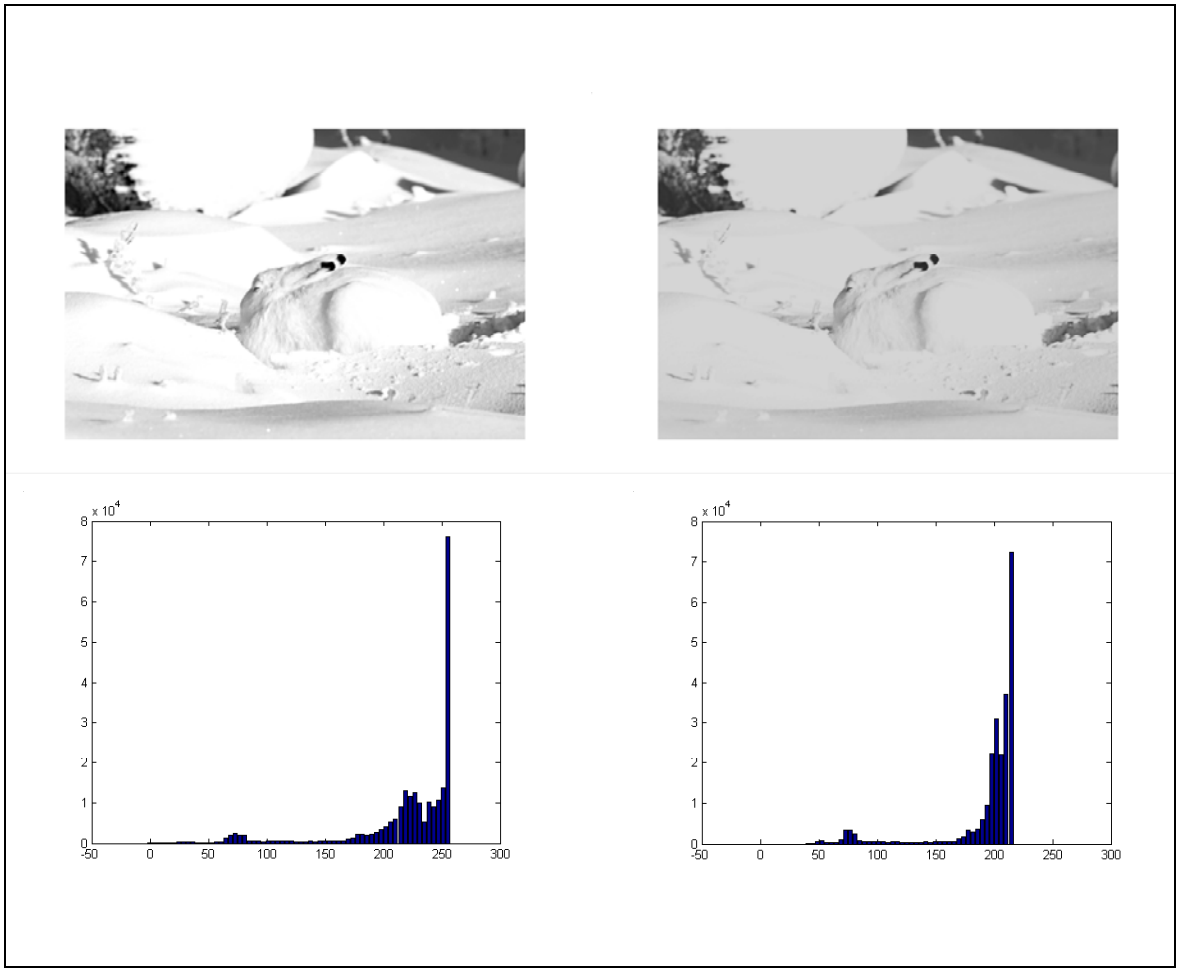

Figure 16. The original (left) and the normalized (right) arctic hare and their corresponding histograms (courtesy of R. E. Barber, Barber Nature Photography).

With the average image, the scheme seems to perform quite well. The use of normalization is optional and drives our attention to the trade-off between transparency and performance.

E. DECISION MAKING DEVICE

For the decision making device, as explained in the previous chapter, we used the correlation coefficient, ρ, between the original watermark, W, and the recovered object, W_r. where ρ was derived from equation 4.5. In the literature there have been several works that use, instead, a different formula (equation 5.4) for the derivation of ρ:

$$\rho = \frac{\sum_i \sum_j W(i,j) Wr(i,j)}{\sqrt{\sum_i \sum_j [W(i,j)]^2 \cdot \sum_i \sum_j [W(i,j)]^2}}, \tag{5.4}$$

References [22], [23], [31], [35] use equation 5.4 or variations of it, which all have in common that there is no participation of the recovered object element, W_r, in the normalizing denominator.

We claim that equation 4.5 (also used in references [33] and [34]) is a more accurate approach. By using 5.4 it is possible that we obtain values of ρ beyond the range [0, 1]. Equation (5.4) can be rewritten as:

$$\rho = \frac{\sum_i \sum_j W(i,j) Wr(i,j)}{\sum_i \sum_j [W(i,j)]^2}. \tag{5.5}$$

Let us consider the special case where W and W_r are exactly the same except for one pixel, say (κ,λ), where $Wr(\kappa,\lambda) > W(\kappa,\lambda)$. Clearly, in this case, the numerator becomes greater than the denominator and thus ρ exceeds 1.

This means that setting the threshold T to 1 does not guarantee that only perfectly retrieved watermarks will pass the evaluation test of the decision making device. In other words the normalization is not correct, and we obtain erroneous impressions leading to incorrectly setting of the threshold value.

F. ERROR CORRECTION CODING

In an attempt to reduce the number of errors that our scheme suffered after attacks (quantization), we tried to implement Error Correction Coding in our algorithm. The 7/15

BCH code capable of correcting up to two errors ([36]) was selected because it was more convenient in terms of overhead and correction capability.

We numbered the watermark pixels p, from 1 to L ($[p_1,p_2,\ldots,p_L]$), starting from the top left corner and sweeping the watermark row-wise from left to right and from top to bottom. From each pixel we encoded the seven most significant bits (leaving out the LSB). These are the bits that contribute more to the determination of the pixel value which in turn affects the result of the correlation of the recovered watermark with the original watermark at the decision making device. The smaller the number of errors in the most significant bits, the higher the correlation between the recovered and the original watermark. Therefore it is very important that these bits remain free of errors.

For every watermark pixel that is encoded (actually its seven most significant bits), we get an 8-bit (or one pixel) overhead from the code. In essence after encoding an $M\times N$ (=L) watermark we obtain an extra $M\times N\times 8$ bits or $M\times N$ (L) pixels that we need to accommodate. The overhead bits form pixels that fill first columns and then rows to the right and bottom of the image. When the overhead pixels are exhausted and the expanded watermark is not a perfect rectangle, we zero pad the remaining bit positions in order to get an $(M+e)\times (N+e)$ watermark, where e is the number of extra rows and columns that were added to the original.

This is illustrated in figure 17. There, one can see the $[p_1,p_2,p_3,\ldots,p_\kappa,p_\lambda,p_\mu,\ldots,p_{L-2},p_{L-1},p_L]$ initial pixels, the respective overhead bits, and how they shape the watermark.

In Appendix A we show the results of the implementation of the ECC.

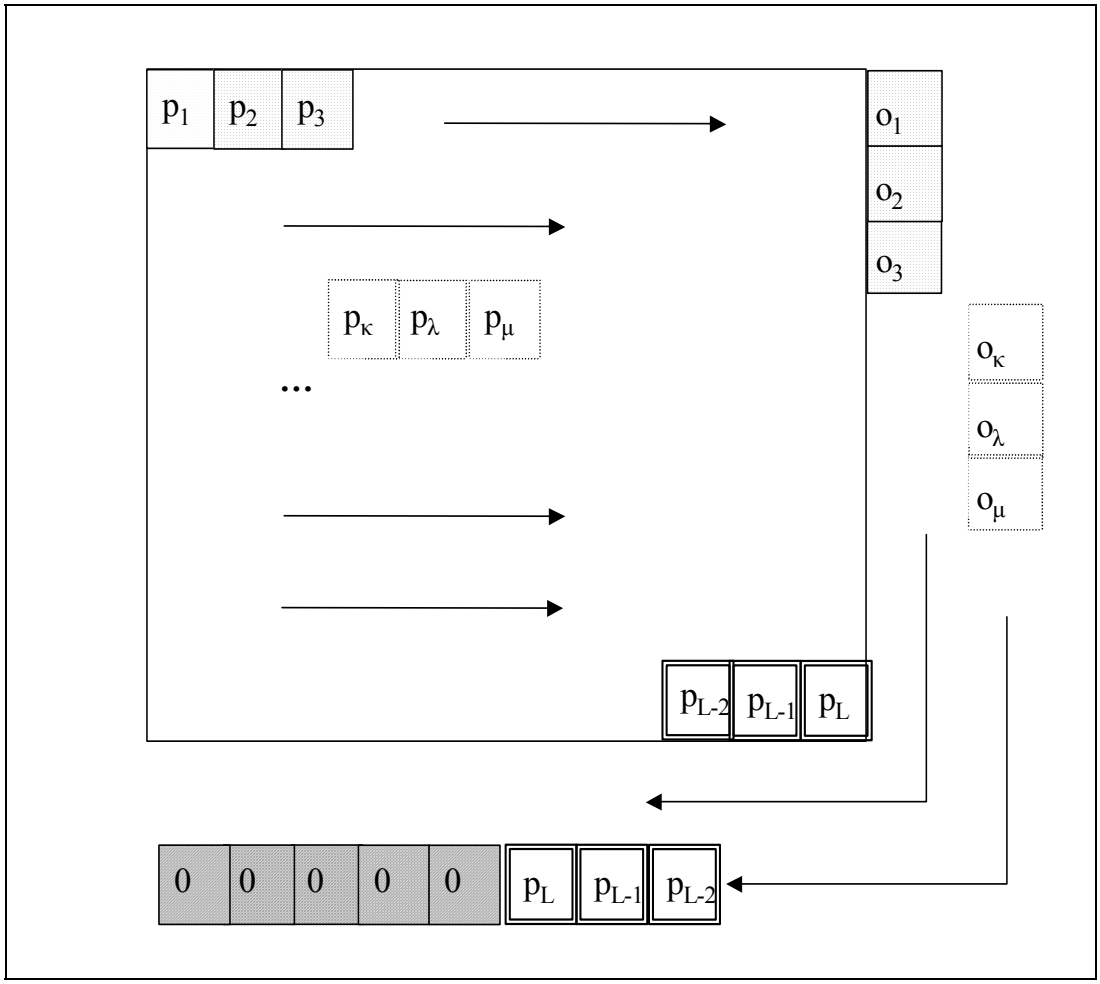

Figure 17. The concept of expanding the watermark after BCH coding, where the pixels p, of the watermark are numbered from 1 to L, and o are the overhead bits.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. RESULTS

Part of the results presented here are also included in reference [37].

A. TESTED IMAGES AND WATERMARKS

1. Images

In this research we used several images bearing different visual and spectral characteristics. The classic images (Lena, figshingboat etc) were primarily used along with some others. There was however interest in conducting experiments with images that had some particular spectral or visual characteristics, which could not be found in the regular images. Therefore we produced a set of what we called artificial images that fitted our needs.

a. Regular (Non-synthetic) Images

We used six images that are shown in figure 18. The distribution of their pixel values is shown in figure 19 where we can see their different characteristics. *Lena*, *peppers* and *fishing boat* cover a broad portion of the allowed range and have various peaks. *Arctic hare* and *fishing boat* have narrower histograms with one large peak, and *New York* exhibits a rather uniform distribution of its pixel values.

b. Artificial (Synthetic) Images

The four artificial images (figure 20) may be grouped into two pairs. *ImageB* and *imageSB* form the first pair, while *imageR* and *imageU* form the second one. Both images of the first pair contain only four different levels of grayscale. In *imageB* we have four large 256×256 blocks. On the other hand *imageSB* is divided into a large number of similarly arranged smaller blocks of 4×4. Thus the two images have exactly the same histogram but different visual and spectral characteristics. Similarly in the case of *imageR* and *imageU* they both contain the whole range of grayscale shades but again they have different visual complexity.

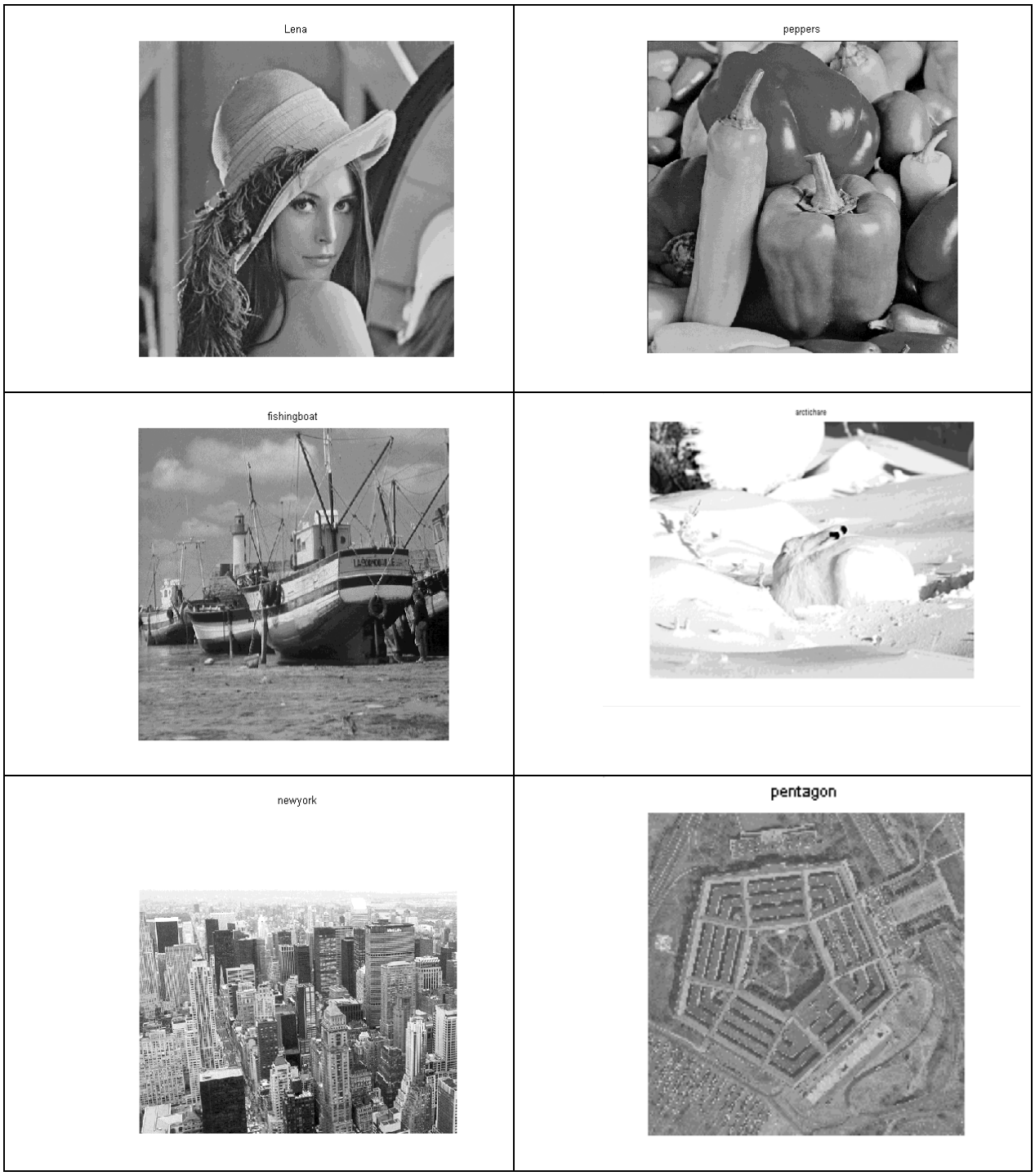

Figure 18. The six regular images that were used in the research.[1]

[1] The image *arctic hare* is courtesy of Robert E. Barber, Barber Nature Photography. The image *New York* is courtesy of Patrick Loo, University of Cambridge. All other non-synthetic images used in this research are courtesy of the Signal and Image Processing Institute at the University of Southern California.

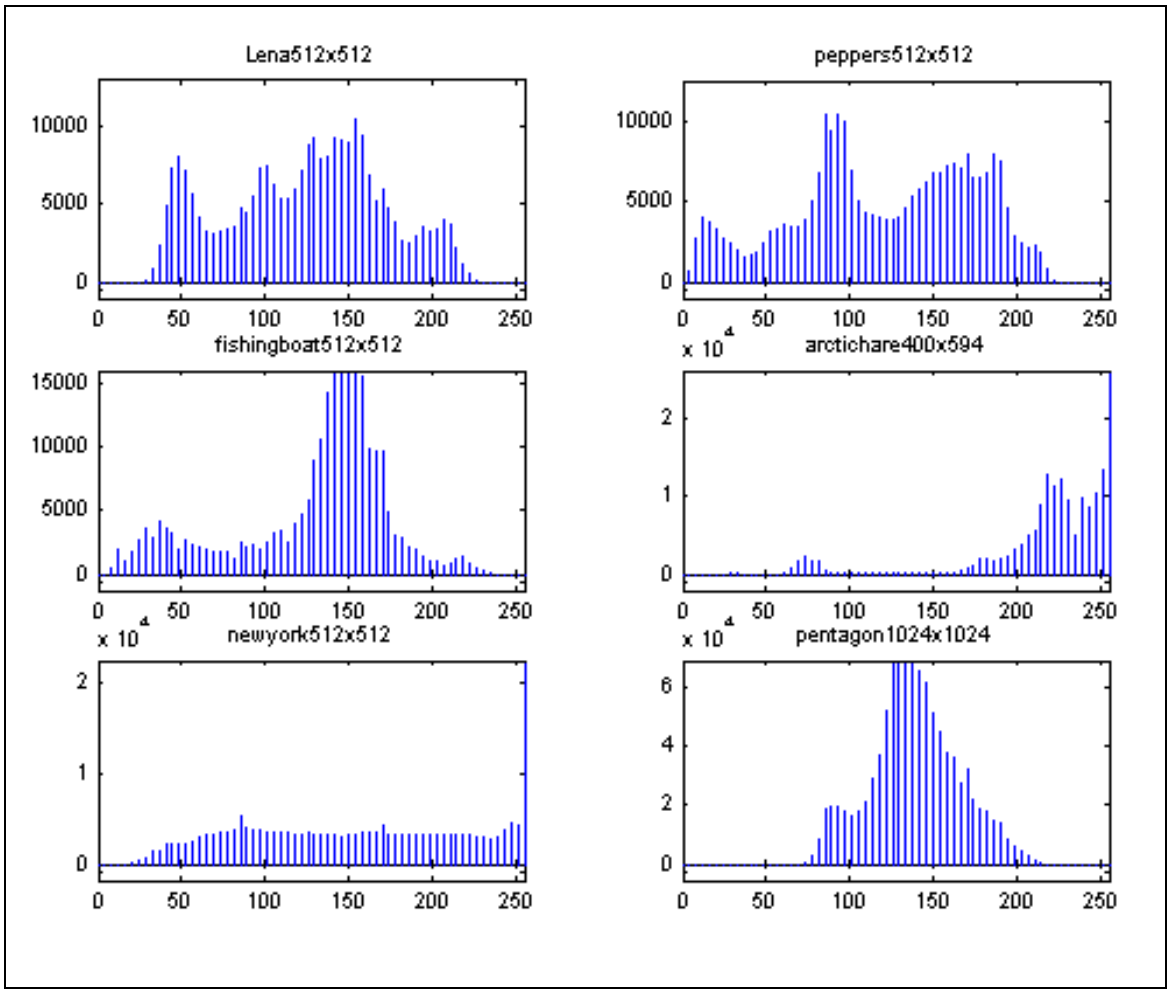

Figure 19. The histograms of the regular images.

2. Watermarks

In the course of our research we first had to decide what watermarks we should use. In the literature different types of watermarks have been proposed. The type of watermark depends on the general concept that we have adopted for our watermarking scheme. One of the most fundamental questions is whether the watermarking algorithm is *open*, or its details are kept secret. In the latter case any pattern, however simple and abstract, may be used as a watermark. In the former case however, a key is required to take care of the security issues. One of the most common suggestions among researchers is a random watermark, which is also a security component. The watermark itself is a key that an intruder is not aware of, and thus cannot verify its existence or removal.

There may be cases however, where a perceptual watermark may be preferable. Such a watermark might be more appealing for commercial purposes as it contains a visually recognizable pattern, such as a copyrighted logo. This implies that either the algorithm should be kept secret or that a unique key should be used for protection.

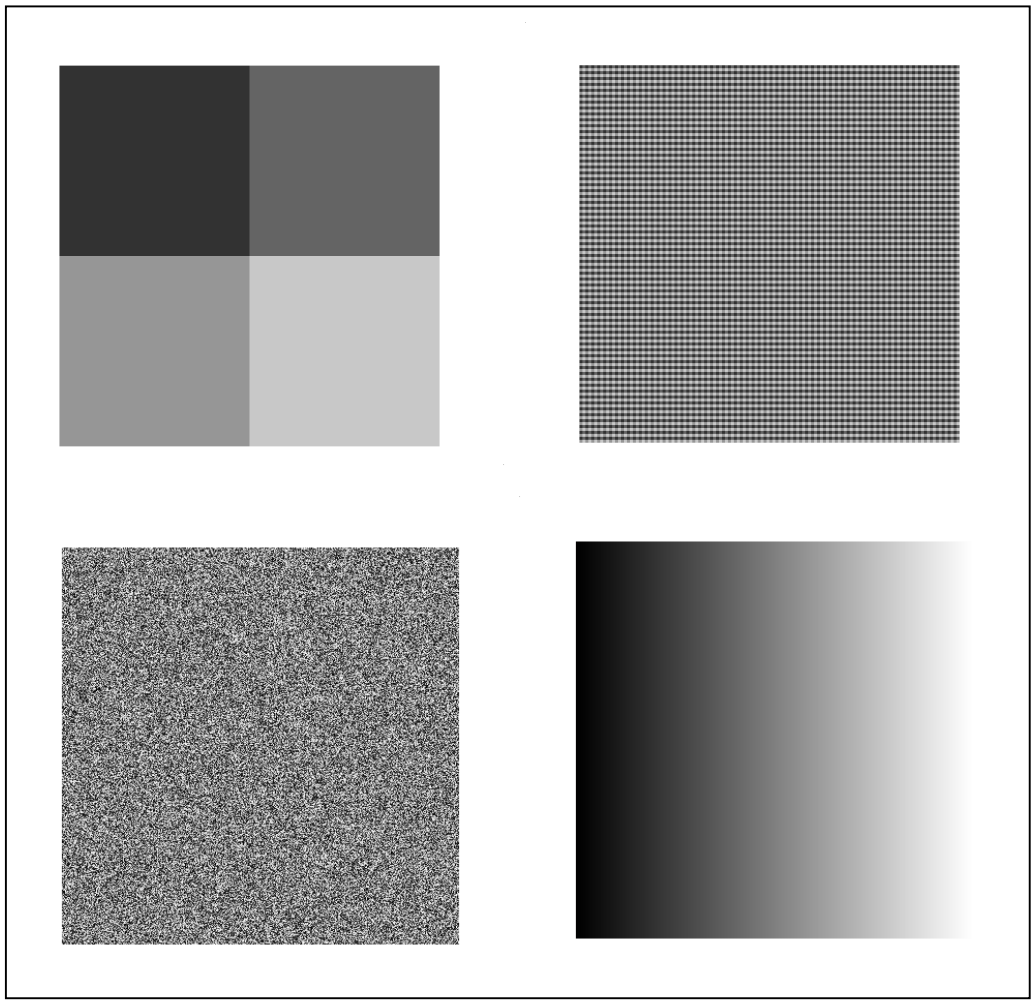

Figure 20. The four artificial images: *imageB* (top left), *imageSB* (top right), *imageR* (bottom left), and *imageU* (bottom right).

a. Watermark Selection

In our research we tried to test several different cases and this justifies the selection of our watermarks (figure 21). The watermark *stripes* is a simple visual pattern and is used as a perceptual watermark. *NPSlogo* is also a perceptual watermark that also has some random characteristics. Finally, *randWm* is a watermark whose pixels are

randomly chosen and follow a uniform distribution. All three watermarks are encoded with 8-bit grayscale in the range ([0, 255]), with dimensions 64×64. We decided to select gray scale watermarks although this imposes higher burden to the marked image in terms of embedding capacity requirements. Grayscale allows for a much larger set of possible keys, thus, providing better security and at the same time allows for more complicated perceptual watermarks.

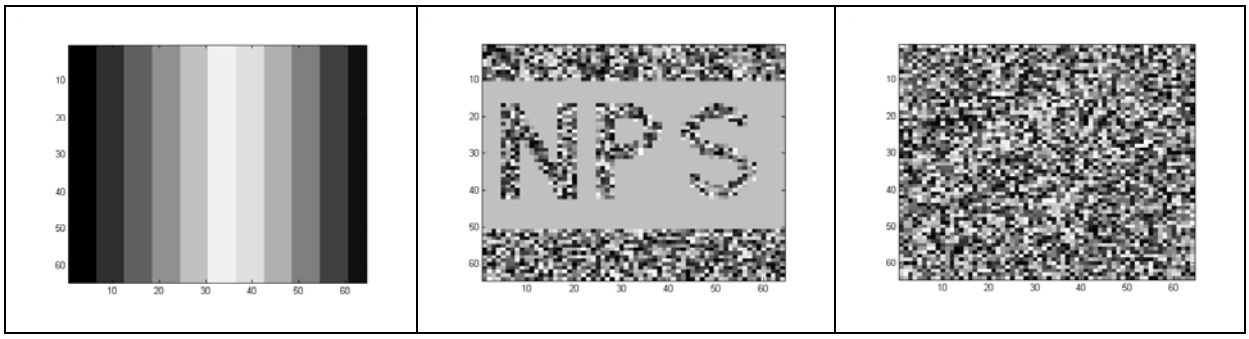

Figure 21. The three used watermarks: *stripes* (left), NPSlogo (middle) where everything except the letters' background is random, and a *randWm* (right) with all pixels uniformly distributed in the range [0, 255].

B. TESTING THE NON-UNIFORM ALGORITHM

1. Transparency

Using the algorithm of Chapter IV with no normalization, and images with 8-bit accuracy, we obtain very good performance in terms of the transparency of the watermark. We can appreciate the results from figures 22 and 23, where we show the marked image next to the original one to allow comparisons. All these examples are produced with the *NPSlogo* watermark, α=0.1, *xstart*=4, *es*=2. If the transparency achieved is evaluated as insufficient, one can make appropriate adjustments to α and *xstart* to satisfy his own requirements.

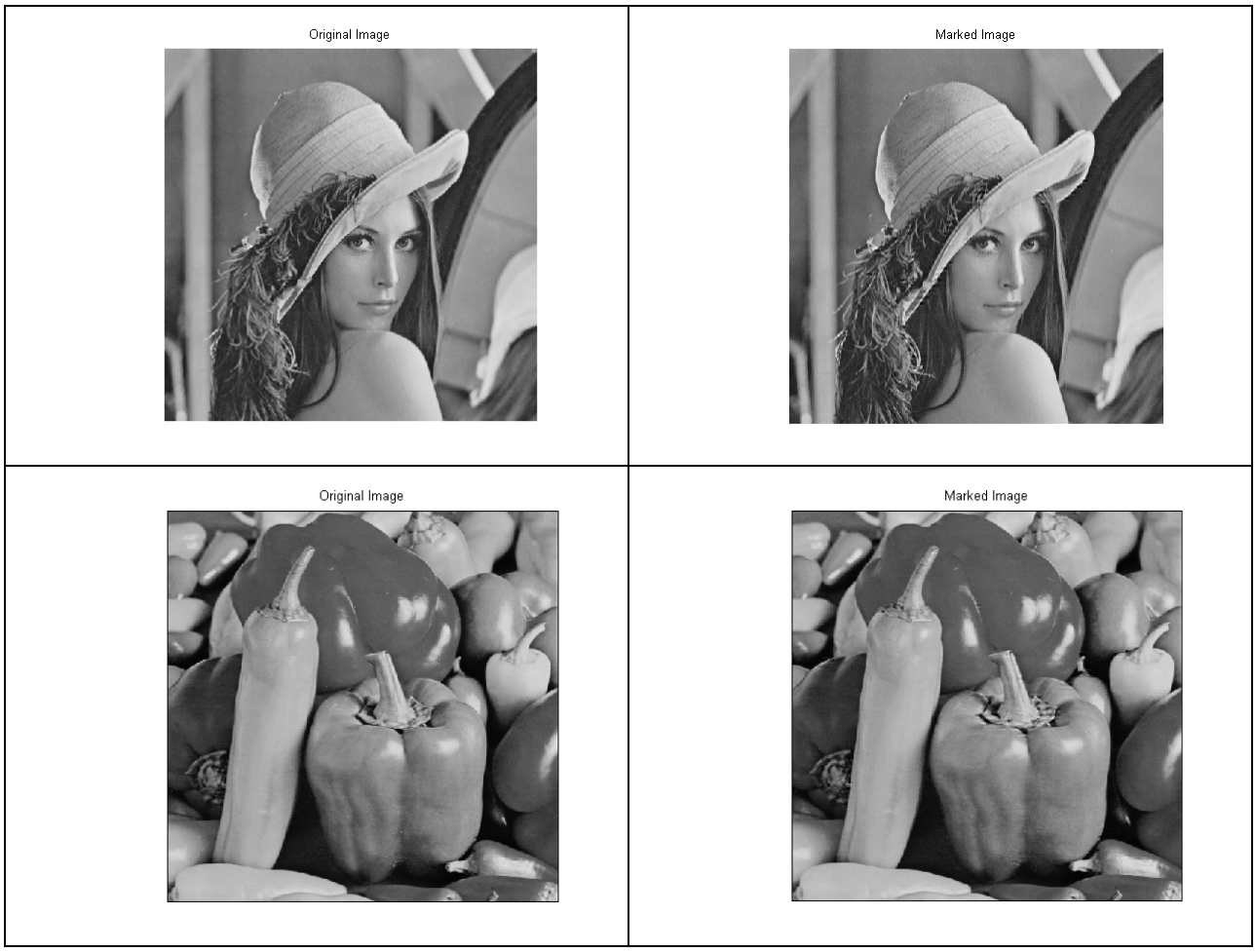

Figure 22. Original and marked (*NPSlogo*) *Lena* (top) and *peppers* (bottom), with α=0.1, x*start*=4, *es*=2. All images are of type uint8 (courtesy of the Signal and Image Processing Institute at the University of Southern California).

Figure 23. Original and marked (*NPSlogo*) *arctic hare* (top) and *New York* (bottom) with α=0.1, *xstart* =4, *es*=2. All images are of type uint8. (*arctic hare* is courtesy of R. E. Barber, Barber Nature Photography, *New York* is courtesy of P. Loo, University of Cambridge)

2. Watermark Recovery from Marked Image

The marked image is of the MATLAB type "unsigned integer with 8 bits" (uint8). Should the system work perfectly, the decoder would normally produce a recovered watermark, which would be identical to the original, since we do not apply any distortions. However the uint8 type is an integer number in the range [0 255] and therefore has finite accuracy (8 bits). After embedding, the IDCT produces non-integer results that are rounded to the nearest integer, and therefore some noticeable distortion is present (figure 24). This means that the correlation coefficient ρ, is in most cases lower than 1 even when the marked image has not been tampered with.

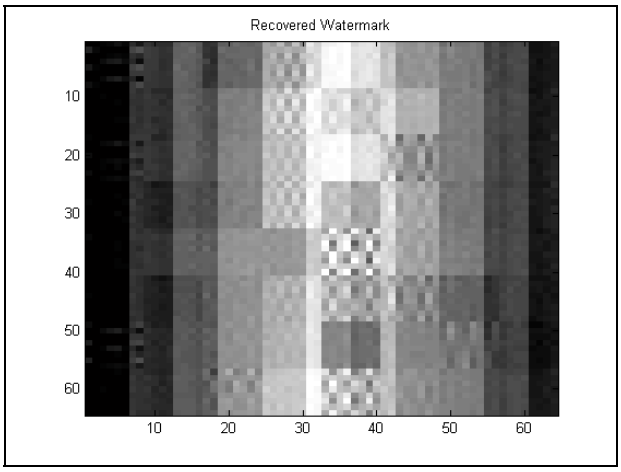

Figure 24. Recovered watermark from the marked *arctic hare* of figure 23.

3. Performance after Quantization

The performance of the algorithm under quantization varies with the frequency band that is selected for the embedding of the watermark coefficients. Figures 25 – 28 show that as we embed in higher frequency coefficients (larger *xstart*), there is a general tendency that the performance becomes poorer (smaller ρ). This is expected, since the higher frequency coefficients are subjected to severe quantization and therefore any small amount of watermarking information contained in these coefficients is essentially eliminated. However we also see that the type of the watermark also matters. A watermark with more random elements (*NPSlogo* as opposed to *stripes*) produces considerably worse results. This can be explained as follows: A simple perceptual pattern like *stripes* has some large coefficients in the lower frequencies and most of the remaining DCT coefficients are zero. The high frequency coefficients are severely quantized and after quantization the watermark information is truncated to zero. If the watermarking information was zero anyway, then the quantization has essentially no effect and therefore the performance is almost the same regardless of the frequency band of the embedding. This train of thought is also verified by the images *imageB* and *imageSB* (figure 27). In each 8×8 block the former contains only a DC offset and all the rest of the coefficients are zero. The latter however, has non-zero coefficients up to the middle frequencies. Therefore for values of *xstart* up to the middle frequencies the performance of these two artificial images is different. From the point where the coefficients become zero and on, their performance becomes identical. This observation

applies only with the watermark *stripes* because the *NPSlogo* contains random pixels that vary considerably for different iterations.

For smaller embedding size *es*, we expect that the transparency of the watermark increases, since the distortion applied to each block is considerably smaller. Additionally, with smaller *es*, the results reveal a tendency for slightly higher ρ (figures 29, 30, 31).

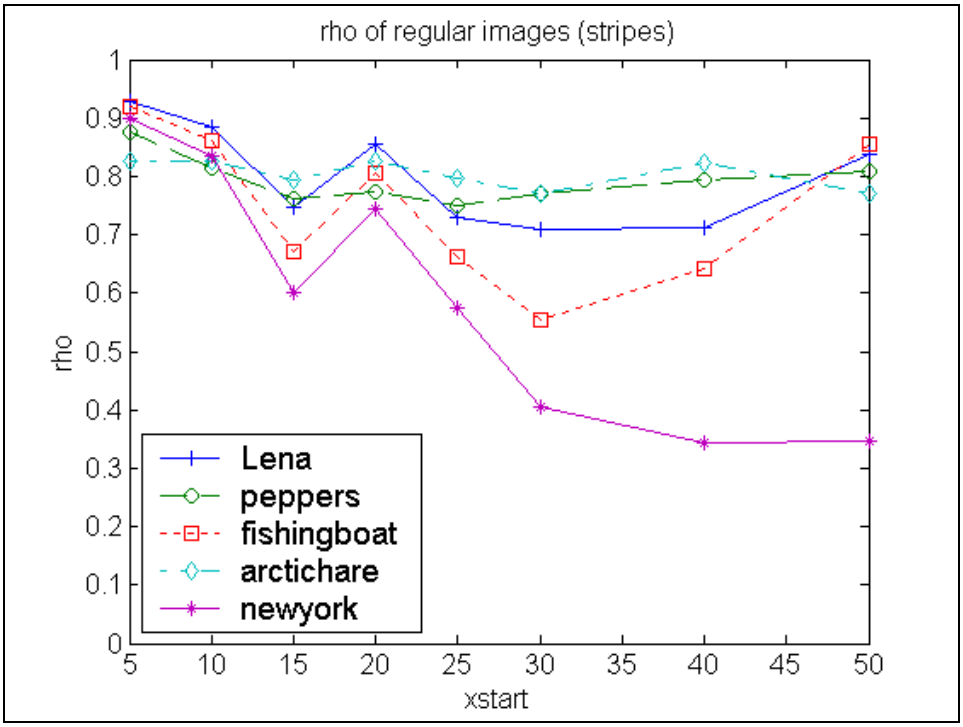

Figure 25. ρ for the regular images with *stripes* and α=0.1, *es*=2.

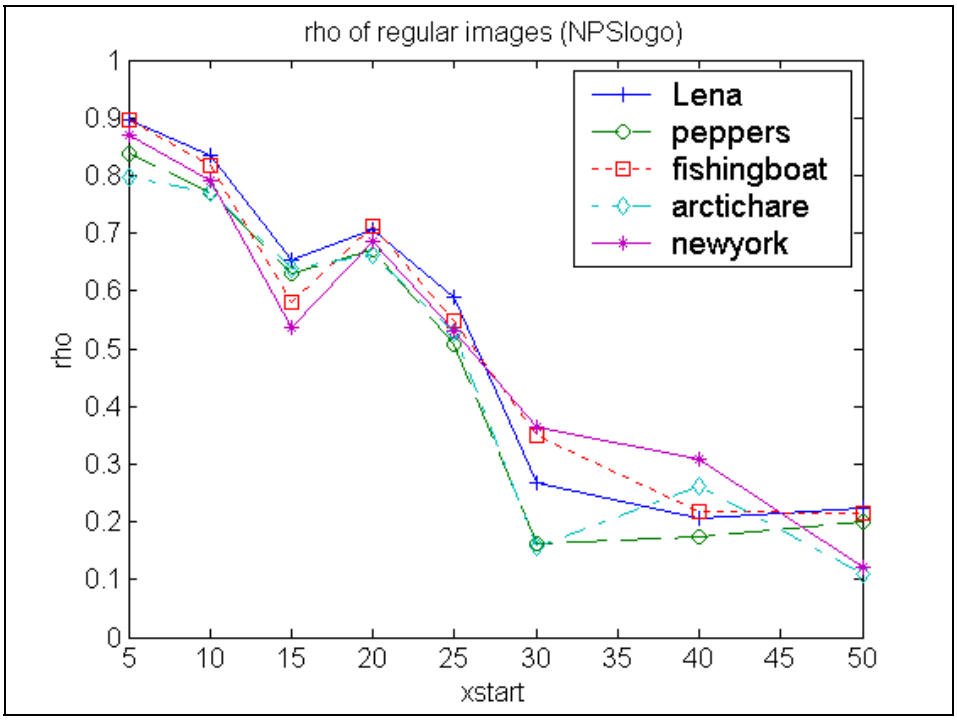

Figure 26. ρ for the regular images with *NPSlogo* and α=0.1, es=2.

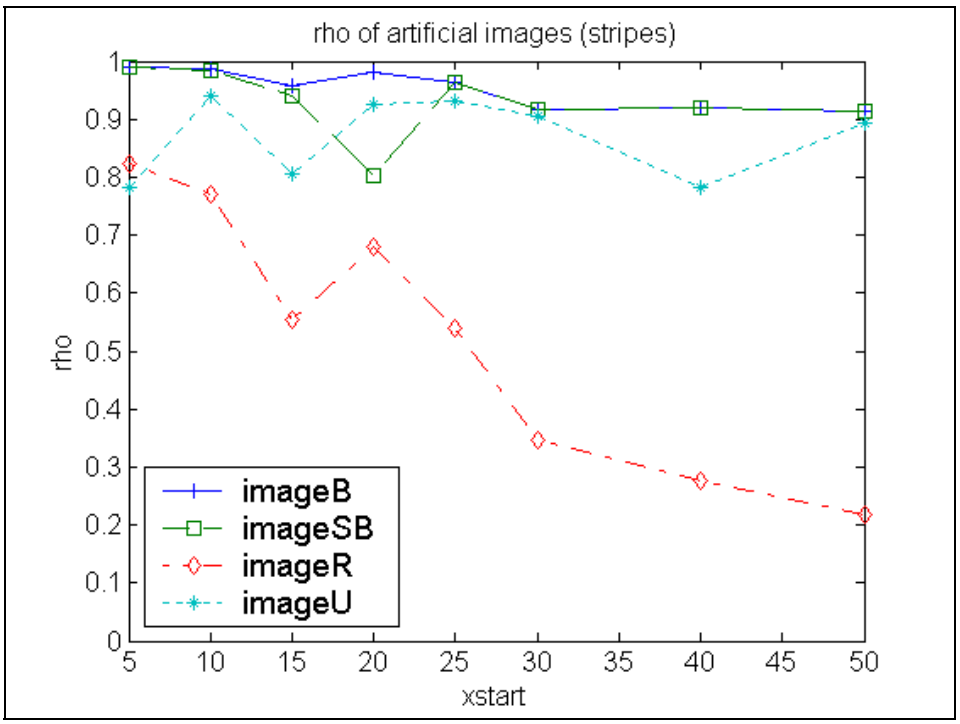

Figure 27. ρ for the artificial images with *stripes* and α=0.1, *es*=2.

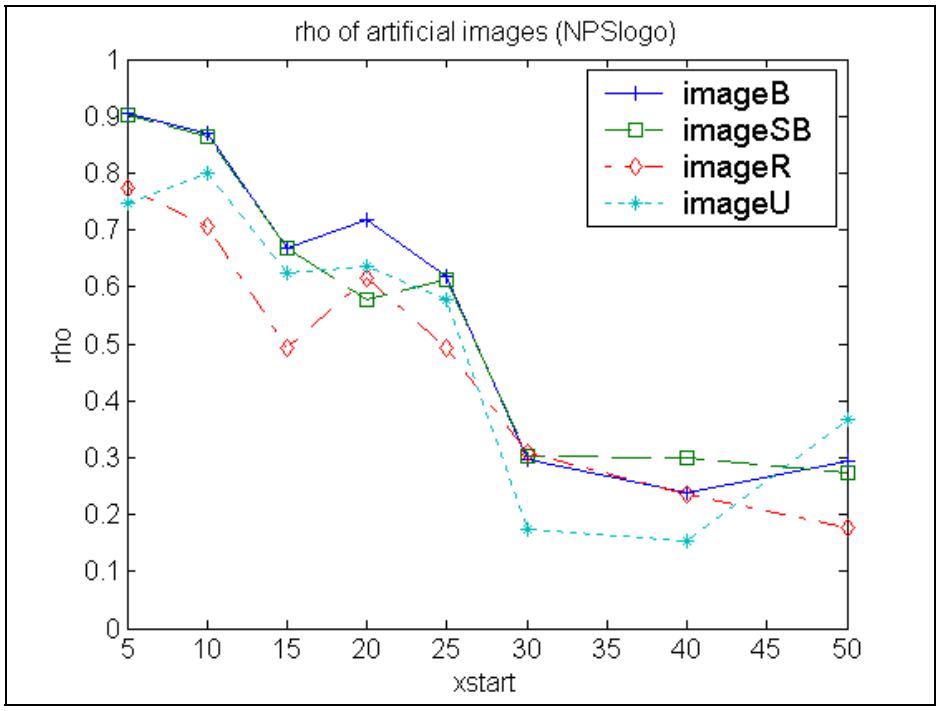

Figure 28. ρ for the artificial images with *NPSlogo* and α=0.1, *es*=2.

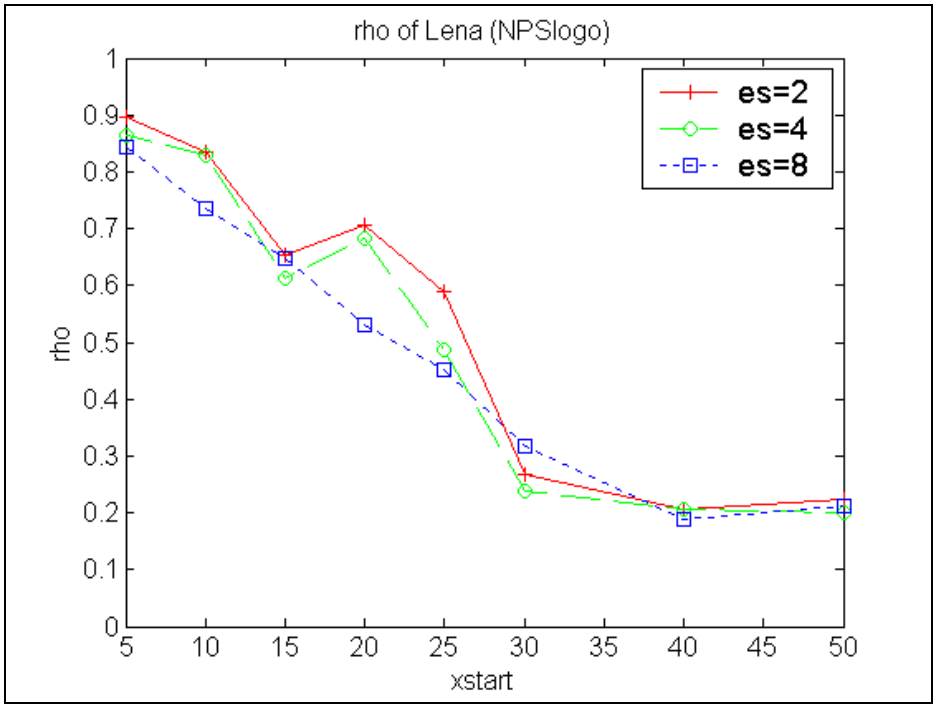

Figure 29. ρ for *Lena* with various embedding sizes and *NPSlogo*, α=0.1.

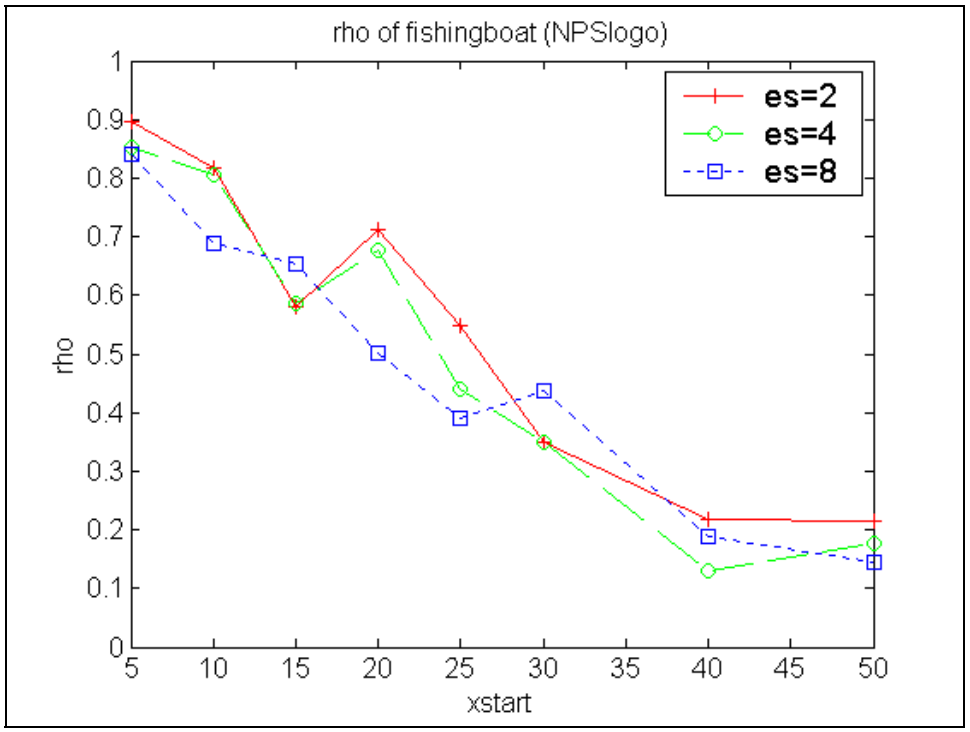

Figure 30. ρ for *fishing boat* with various embedding sizes and *NPSlogo*, α=0.1.

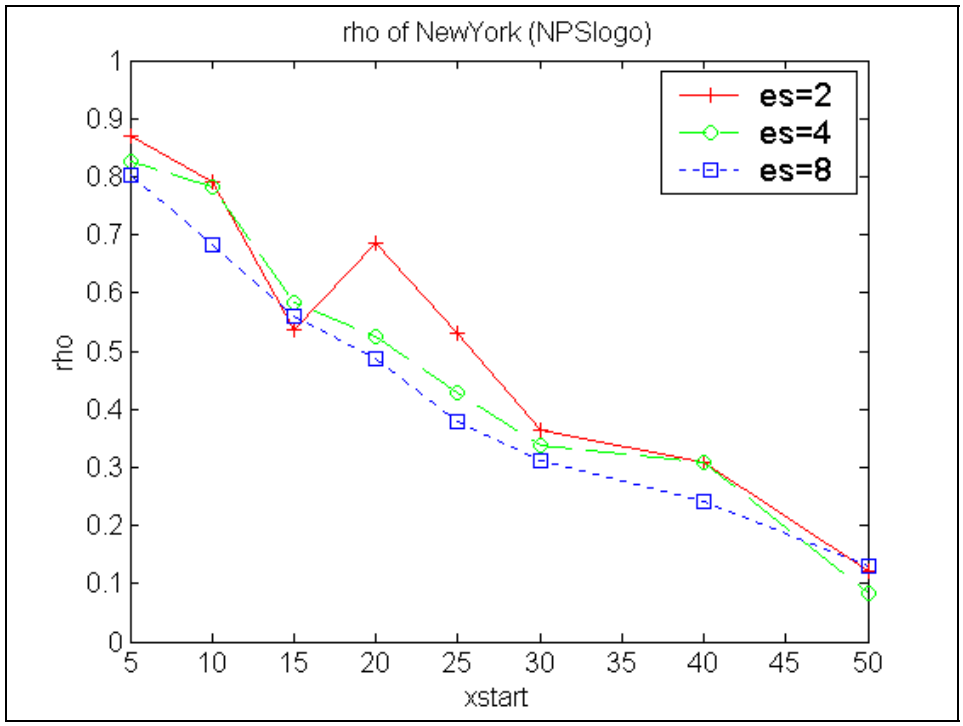

Figure 31. ρ for *New York* with various embedding sizes and *NPSlogo*, α=0.1.

Figures 25 – 28 show that the embedding setup can be adjusted accordingly to produce values of ρ very close to 1. This means that an appropriate threshold T can be chosen depending on our P_{FA} and P_D requirements (see section IV.C).

4. Robustness to Cropping

The performance of our algorithm to cropping varies with the images marked, and there are cases where it can be really exceptional. The results are shown in tables 7 and 8. As expected, the performance improves as the size of cropping increases. However there should be a point where the size of the cropping becomes too large for the algorithm to handle. From that point on, too many essential watermark coefficients embedded towards the Center of Interest, are cropped out, making it impossible for the algorithm to perform sufficiently. At this point however the image is so severely cropped that becomes useless for any potential adversary (figure 32).

Table 7. Performance of the Non-uniform Algorithm against Cropping (*NPSlogo*, α=0.1, *xstart*=4, *es*=2).

Image	**Maintained pixels after cropping (initially 512×512)**	**ρ without CIPF**	**ρ with CIPF**	**Improvement**
Lena	(50:460,50:460)	0.5121	0.8524	66.45%
New York	(50:460,50:460)	0.0544	0.2689	394.3%
fishing boat	(50:460,50:460)	0.2612	0.3855	47.58%
peppers	(50:460,50:460)	0.0671	0.3810	467.81%

Table 8. Performance of the non-uniform algorithm against cropping (*NPSlogo*, α=0.1, *xstart*=4, *es*=2).

Maintained pixels after cropping (initially 512×512)	***Lena***			***fishing boat***		
	ρ without CIPF	**ρ with CIPF**	**Improvement**	**ρ without CIPF**	**ρ with CIPF**	**Improvement**
(11:502, 11:502)	0.5704	0.9172	60.79%	0.5749	0.8467	47.27%
(31:482, 31:482)	0.4143	0.7558	82.42%	0.2728	0.5595	105.09%
(51:462,51:462)	0.3038	0.5575	83.50%	0.1663	0.3522	111.78%
(71:442, 71:442)	0.2253	0.3681	63.38%	0.1580	0.2390	51.32%
(91:422,91:422)	0.1906	0.2906	52.46%	0.1081	0.1540	42.46%
(111:402, 111:402)	0.1384	0.1838	32.80%	0.0921	0.1223	32.79%

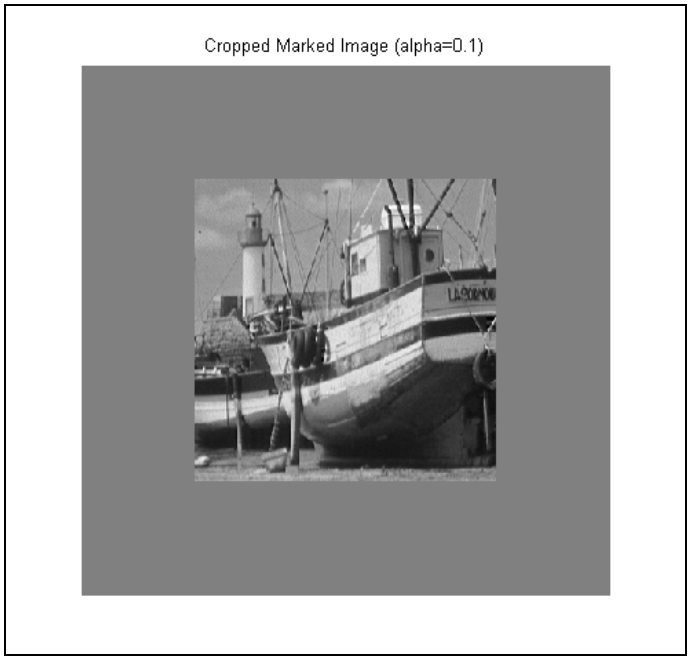

Figure 32. Cropped *fishing boat* with remaining pixels [111:402, 111:402] from a 512×512 image (courtesy of the Signal and Image Processing Institute at the University of Southern California).

C. SELECTION OF THE WEIGHTING FACTOR

The weighting factor α, used in the embedding algorithm (section IV.B) also affects the system's performance. Figures 33 and 34 show that New York and Lena behave similarly, both in the case where the image is quantized and in the case where it is not.

Examining the performance, ρ, of the system with *New York* for various values of α when the marked image is of type uint8, we notice that, in case of no attack to the marked image, we get a maximum of ρ at α=0.1, with the values varying slightly between 0.9477 (for α=0.6) and 0.9934 (for α=0.1). If quantization is applied, it is reasonable that the performance would be different. Up to a certain value of α, we have a dramatic performance improvement. As α still increases, the amount of improvement is reduced and the performance becomes essentially unchanged. The improvement stops for α around 0.3. At this value of the weighting factor however, it is possible that the effects of the embedding in the marked image are already visible (figure 35). The results after quantization can be significantly worse if we choose to embed in higher frequency components (subjected to severe quantization). In the scheme we proposed in Chapter IV we use α=0.1 which appears to give the best performance if no quantization is used.

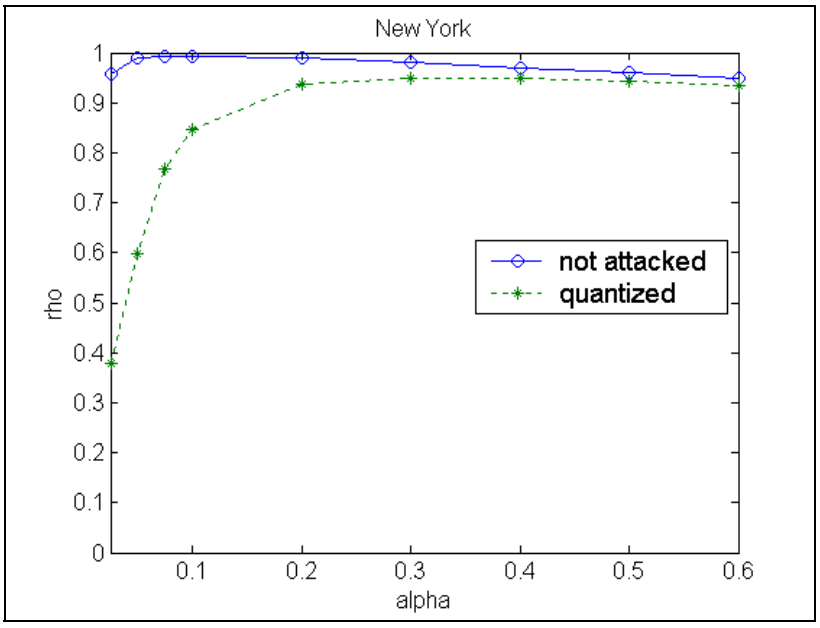

Figure 33. Performance measured on the marked *New York* image (uint8) for various values of α (watermark: *NPSlogo*, *xstart*=4, embedding size=2).

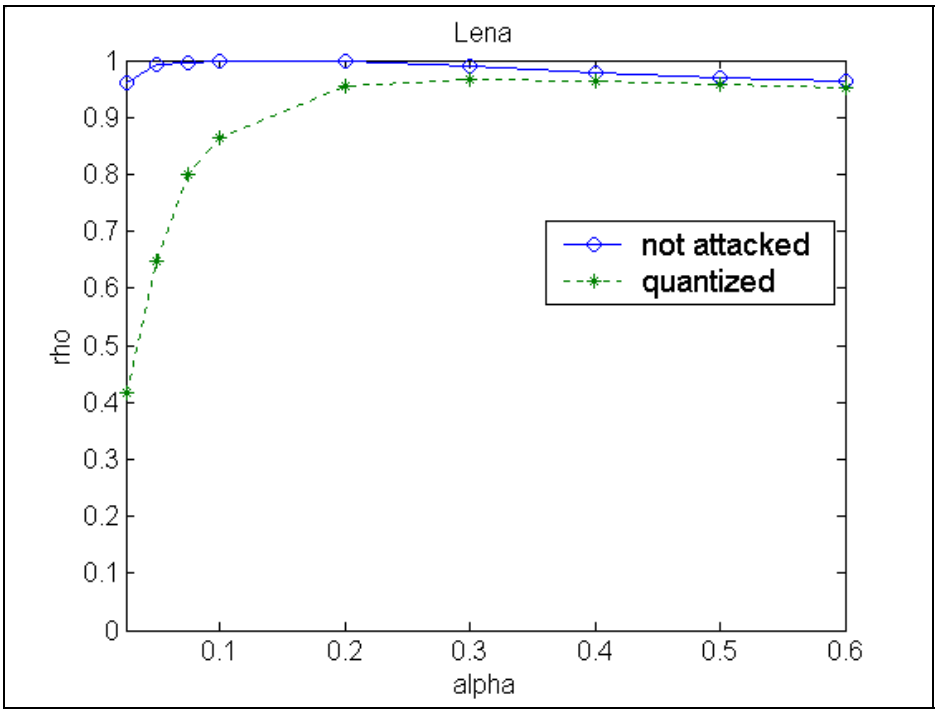

Figure 34. Performance measured on the marked *Lena* image in uint8 for various values of α (watermark: *NPSlogo*, *xstart*=4, embedding size=2).

Figure 35. *Lena* marked with *NPSlogo* and *xstart*=4, *es*=2. The distortion at α=0.3 is clearly visible (courtesy of the Signal and Image Processing Institute at the University of Southern California).

VII. CONCLUSION

A. SUMMARY

In this thesis we studied the use of the DCT in digital watermarking. After a historic overview and a brief presentation of the relevant work that has been conducted by other researchers (Chapter II), we went through a brief analysis of the theory behind the DCT and the JPEG compression (Chapter III), so that the reader would obtain all the necessary background for the comprehension of our work.

In Chapter IV we presented a new non-uniform watermarking algorithm that is based on the Discrete Cosine Transform. The algorithm embeds normally in the lower frequency components (this case proved to give better results), and achieves sufficient transparency of the watermark, but also robustness against quantization (i.e. JPEG compression) and cropping (Chapter VI). Finally, some other parts of our research worth mentioning are discussed in Chapter V.

B. SIGNIFICANT REMARKS

The performance of Discrete Cosine Transform - based digital watermarking still needs further investigation. The development of the JPEG 2000 standard and how this will affect the domination of JPEG should be taken into account for future research in the watermarking area.

There are qualities of the grayscale watermark that affect the recovery process. A simple perceptual (*stripes*) watermark behaves differently from a highly random one (*NPSlogo*). The randomness imposes an additional obstacle to the decoder and reduces the performance of the decision making device especially in the higher frequencies.

In the proposed scheme we used a unique metric for measuring the relative capacity of each image block to receive watermark information without perceptual distortion of the overall image. In addition another metric is used (CIPF) to defeat cropping attacks. The combination of the two metrics is used to prioritize the image blocks and determine the watermark coefficients that will be embedded in each one of them. The achieved transparency appears to be sufficient but the evaluation is rather subjective, based only on observation.

The scheme responds quite well to quantization allowing for the determination of a threshold T according to our P_{FA} and P_D requirements. As we embed towards higher frequency coefficients the performance becomes poorer because of the severe quantization in the higher frequency bands. Decreasing the *embedding size* appears to slightly contribute to improvement of the overall performance. Additionally ρ is also affected by variations of the weighting factor α. The experiments show that there is a trade-off because as we increasing α, we improve ρ but also the transparency is negatively affected. We should bear in mind that there are certain limits suggested by data where increasing α is meaningful. Beyond these limits the watermark correlation coefficient is not further improved and the transparency is essentially degraded.

The performance of the proposed scheme against cropping attacks, ranges from mediocre to exceptionally good results, depending on the input data. Improvement of the correlation coefficient exceeding 100% is frequently observed depending on the tested image and the amount of cropping.

C. FUTURE WORK

Embedding the coefficients in a way that will take full advantage of the human visual system's characteristics is a big goal of the watermarking community. Further research in this subject is required to investigate the possibility of incorporating our metric to the Just Noticeable Difference (JND) models that have been proposed.

In addition, in this work the transparency of a watermarking algorithm is judged by the subjective decision of independent observers. The possibility of developing a formal model for the evaluation of the transparency may be investigated. However this task is not trivial. A simple correlation test between the original and the marked image would not work. This would detect any differences between the two but cannot tell if these differences occur in a visually perceptual manner. A JND model could be used as the basis for the evaluation of the transparency ([24]), but then it should not also be used in the embedding model. Otherwise the judgment would be biased and therefore unable to give dependable results.

D. EPILOGUE

The watermarking community is still far from presenting a dominating watermarking scheme. The research in digital watermarking has been dictated by the developments in the digital world and the need for a dependable copyright protection scheme. In other words, the watermarking community has been just following the advances of other technologies. Maybe this is the largest drawback that keeps the watermarkers away from the desired goal. The extreme pace with which the digital technologies are progressing, does not allow sufficient time for the research in the various watermarking areas of interest to mature and produce results. A more independent path may be the secret for the success, and maybe, in the future, the compression algorithms or storage techniques will be following the developments of the watermarking world.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. RESULTS OF THE ECC IMPLEMENTATION

The attempt to incorporate an error correction code in the scheme failed because of the insufficient error correction capability of the code we chose. Investigating the bit error rate between the original and the recovered watermark we see that in general is beyond the 2 bits that the 7/15 BCH code is capable of correcting. The results are shown in figures 36 – 39. The bit error rate, *BER*, calculates the number of errors in bits per pixel (figures 36, 37). To get a better idea about the distribution of errors among the pixels we also used a modified bit error rate metric, the *BERmod*, which is the number of errors in bits per pixel in error (figures 38, 39). This is determined by the total number of bit errors and divided by the number of pixels where errors are detected. For all the tested images the results were very similar and definitely exceeded 3 bits per pixel for both metrics. The *BERmod* plots are almost the same as the *BER* ones, showing that the errors were evenly distributed among the watermark pixels.

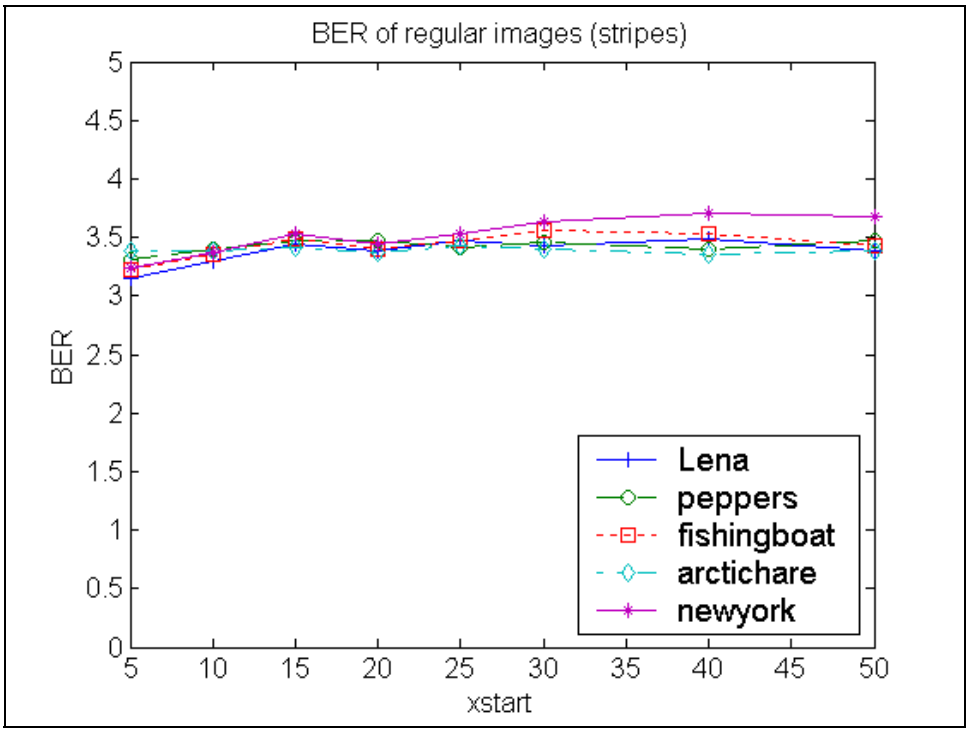

Figure 36. *BER* of regular images with watermark *stripes*.

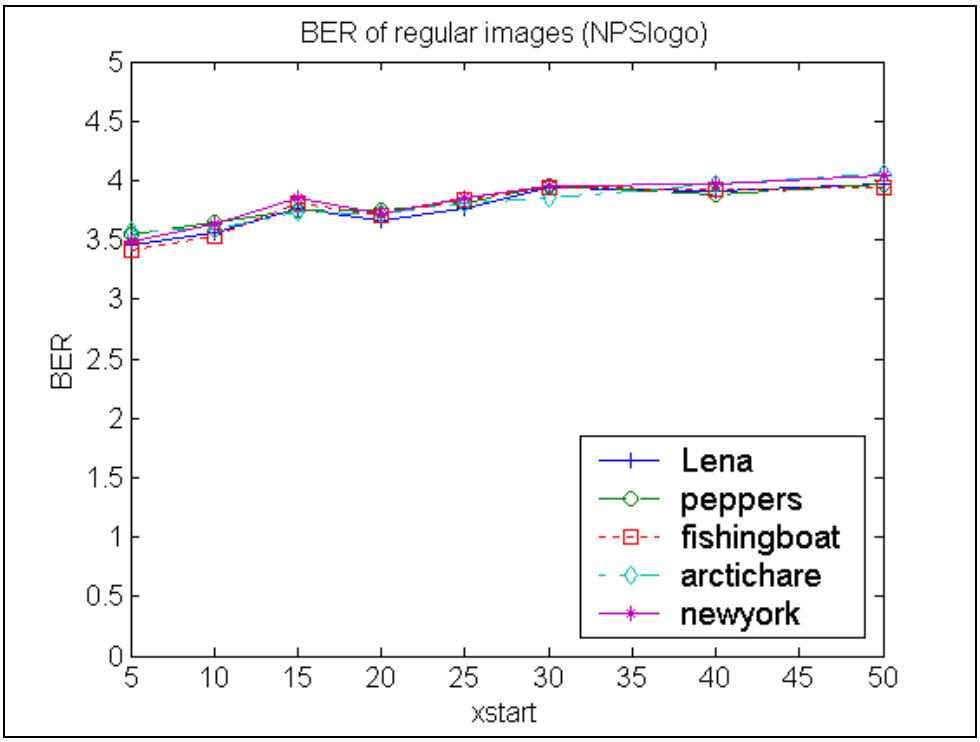

Figure 37. *BER* of regular images with watermark *NPSlogo*.

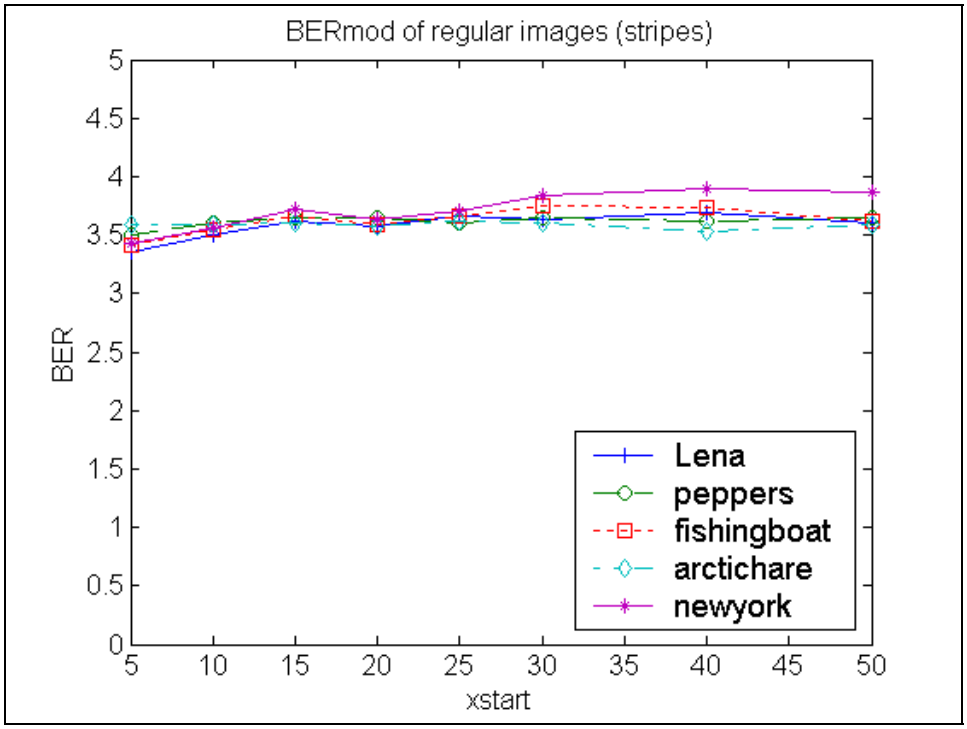

Figure 38. *BERmod* for regular images with watermark *stripes*.

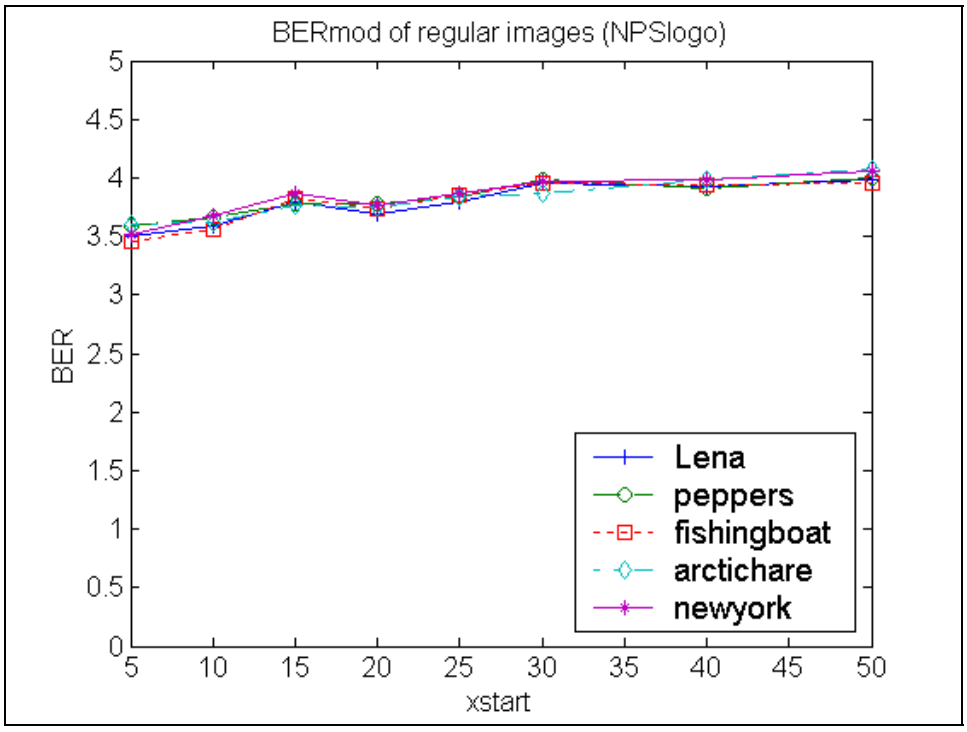

Figure 39. *BERmod* for regular images with *NPSlogo*.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SOFTWARE

```
%*************************************************************************
%*************************************************************************
% Ioannis Retsas
% August 12, 2001
% LAST MODIFICATION: February 4, 2002
% FILE NAME: Encoder
% DESCRIPTION: This is the main encoding file of our watermarking framework.
%*************************************************************************
%*************************************************************************

clear all
delete C:\matlabR12\work\*.mat
disp('Processing...')
%*******************************SETUP SECTION*******************************
I = imageSelectionC; % selecting an image for processing from the gallery
save C:\matlabR12\work\I I
[M,N] = size(I); % M,N are the image dimensions
if ((M/8)/fix(M/8) ~= 1) | ((N/8)/fix(N/8) ~= 1)
    fprintf(1,'The dimensions of the selected image are not multiples of 8\nand
errors will occur;\nTHE PROGRAM IS TERMINATED\N');
    return
end
fprintf(1,'The watermark size is set by default to 64x64;\n');
% Any modification of the size should consider the dimensions of the image and
% the embedding size.
Mw = 64;
Nw = 64;
W = WmTypeC(Mw,Nw);
save C:\matlabR12\work\W W
%------------------------WEIGHTING FACTOR-----------------------------------
alpha = input('Set the weighting factor alpha (recommended value 0.1);\n');
disp('Processing...')
save C:\matlabR12\work\alpha alpha
%-----------------------------XSTART--------------------------------------
start = input('Set the index of the coefficient (1 to 64) where the\nembedding
would start in each block;\n');
while (start <= 0)|(start >= 64)|(start/fix(start)~=1)
```

```
    fprintf(1,'Your choice was either beyond the allowed range or was not an
integer;\n');
    start = input('Try again:\n');
end
disp('Processing...')
save C:\matlabR12\work\start start
%---------------------------EMBEDDING SIZE---------------------------------
fprintf(1,'Set the embedding size (number of watermark coefficients per
block);\n');
length = input('Choose 2, 4 or 8;\n');
while (length ~= 2)&(length ~= 4)&(length ~= 8)
    start = input('Your choice should be 2, 4 or 8; Try again:\n');
end
disp('Processing...')
save C:\matlabR12\work\length length
%-----------------------------CROPPING SETUP-------------------------------
flagCrop = input('For cropping press 1; otherwise press 0;\n');
while (flagCrop ~= 0)&(flagCrop ~= 1)
    flagCrop = input('Your choice should be either 0 or 1; Try again:\n');
end
disp('Processing...')
save C:\matlabR12\work\flagCrop flagCrop
if flagCrop == 1
    leftB = input('Enter the column that will be the new LEFT border of the
Image;\n');
    disp('Processing...')
    rightB = input('Enter the column that will be the new RIGHT border of the
Image;\n');
    disp('Processing...')
    upperB = input('Enter the row that will be the new UPPER border of the
Image;\n');
    disp('Processing...')
    lowerB = input('Enter the row that will be the new LOWER border of the
Image;\n');
    disp('Processing...')
    cropParam = [leftB rightB upperB lowerB];
    save C:\matlabR12\work\cropParam cropParam
end
%-----------------------------QUANTIZATION---------------------------------
flagQ = input('For quantization press 1; otherwise press 0;\n');
while (flagQ ~= 0)&(flagQ ~= 1)
    flagQ = input('Your choice should be either 0 or 1; Try again:\n');
```

```
end
disp('Processing...')
%--QUALITY FACTOR--
if flagQ == 1
    q_jpeg = input('Set the quality factor q_jpeg in the range [1,100];\n');
    while (q_jpeg < 1)|(q_jpeg > 100)|(q_jpeg/fix(q_jpeg)~=1)
        fprintf('Your choice was either beyond the allowed range or was not an
integer;\n');
        q_jpeg = input('Try again:\n');
    end
    disp('Processing...')
end
save C:\matlabR12\work\flagQ flagQ
%--------------------------MARKED IMAGE IN UINT8----------------------------
flag8 = input('For marked image in uint8 press 1; otherwise press 0;\n');
while (flag8 ~= 0)&(flag8 ~= 1)
    flag8 = input('Your choice should be either 0 or 1; Try again:\n');
end
disp('Processing...')
save C:\matlabR12\work\flag8 flag8
%-- IF MARKED IMAGE REAL --> NORMALIZATION
if flag8 == 0
    flagNorm = input('For normalization press 1; otherwise press 0;\n');
    while (flagNorm ~= 0)&(flagNorm ~= 1)
        flagNorm = input('Your choice should be either 0 or 1; Try again:\n');
    end
    disp('Processing...')
    save C:\matlabR12\work\flagNorm flagNorm
end
%*************************************************************************
%**********************PROCESSING SECTION*********************************
%------------------------DCT OF THE IMAGE---------------------------------
Id = double(I); % It is assumed that the image from is in uint8 form [0 255]
T = dctmtx(8);
dctI = blkproc(Id,[8 8],'P1*x*P2',T,T');
save C:\matlabR12\work\dctI dctI
%----------------------DCT OF THE WATERMARK-------------------------------
Wd = double(W);
dctW = blkproc(Wd,[8 8],'P1*x*P2',T,T');
save C:\matlabR12\work\dctW dctW
```

```
clear Wd
%---------------------------EMBEDDING-------------------------------------
dctI = embed8(dctI,dctW,alpha,start,length);
clear dctW
%----------------IDCT OF MARKED IMAGE COEFFICIENTS-------------------------
Im = blkproc(dctI,[8 8],'P1*x*P2',T',T); % scrambled marked image
clear dctI
%--------------------------UINT8 - SNR-------------------------------------
if flag8 == 1
    Im = uint8(Im); % Im is the marked image in uint8
    SNR8 = SNR(Id,double(Im));
    fprintf(1,'SNR of uint8 image, SNR8(dB)=%1.4f\n',SNR8);
else
%-------------------------NORMALIZATION - SNR-------------------------------
    SNRr = SNR(Id,Im);
    fprintf(1,'SNR of real image, SNRr(dB)=%1.4f\n',SNRr);
    Im = Im/255; % reduce Image to range [0 1] plus some distortion caused from
the embedding
    if flagNorm == 1
        n = 3.5;  % selected optimal value
        save C:\matlabR12\work\n n
        Im = 1/pi*atan(n*(Im-1/2))+1/2; % normalization
        SNRnorm = SNR(I,(255*Im));
        fprintf(1,'SNR        of        real,        normalized        image,
SNRnorm(dB)=%1.4f\n',SNRnorm);
    end
end
save C:\matlabR12\work\Im Im
%-------------------------QUANTIZATION----------------------------------------
if flagQ == 1
    if flag8 == 0
        Im = 255*double(Im); % we multiply by 255 to return to the correct
scale
    end
    Imq = qFunc(Im,q_jpeg);
    if flag8 == 0
        SNRrmq = SNR(Id,Imq);
        fprintf(1,'SNR    of    real,    marked    and    quantized    image,
SNRrmq(dB)=%1.4f\n',SNRrmq);
        Imq = Imq/255;
    else
```

```
        Imq = uint8(Imq);
        SNR8mq = SNR(Id,double(Imq));
        fprintf(1,'SNR    of    uint8,    marked    and    quantized    image,
SNR8mq(dB)=%1.4f\n',SNR8mq);
    end
    save C:\matlabR12\work\Imq Imq
end
clear Id
%*************************************************************************
%***********************DISPLAY SECTION*********************************
figure(1)
imshow(I)
title('Original Image')

% figure(2)
% imagesc(W,[0 255]), colormap(gray),
% title('Watermark')

figure(3)
imshow(Im)
title('Marked Image')

% figure(4)
% imshow(Im-double(I)/255)
% title('Difference between Marked & Original')
if flagQ == 1
    imhist(Imq,64)
end

if flagQ == 1
     figure
     imshow(Imq)
     title('Quantized Marked Image')
end
```

```
%*******************************************************************************
%*******************************************************************************
% Ioannis Retsas
% Aug 12, 2001
% LAST MODIFICATION: February 12, 2002
% FILE NAME: Decoder
% DESCRIPTION: This file recovers the Watermark from a marked Image
%*******************************************************************************
%*****************************DATA LOADING**********************************
clear all



load C:\matlabR12\work\indexGr
load C:\matlabR12\work\index
load C:\matlabR12\work\alpha
load C:\matlabR12\work\start
load C:\matlabR12\work\length
load C:\matlabR12\work\flagCrop
load C:\matlabR12\work\flagQ
if flagQ == 1
    select = input('Press 0 to process the marked image; press 1 to process the
quantized, marked image\n');
    while (select ~= 0)&(select ~= 1)
        select = input('Your choice should be either 0 or 1; Try again:\n');
    end
end
disp('Processing...')
load C:\matlabR12\work\flag8
if flag8 == 0
    load C:\matlabR12\work\flagNorm
end
load C:\matlabR12\work\I
load C:\matlabR12\work\W
% load C:\matlabR12\work\Key
[Mw,Nw] = size(W);
if flagQ == 1
    if select == 0
        load C:\matlabR12\work\Im
        Itest = Im;
        clear Im
```

```
    elseif select == 1
        load C:\matlabR12\work\Imq
        Itest = Imq;
        clear Imq
    end
else
    load C:\matlabR12\work\Im
    Itest = Im;
    clear Im
end
Itest = double(Itest);
%*************************************************************************
%*******************************CROPPING**********************************
if flagCrop == 1
    load C:\matlabR12\work\cropParam
    if flag8 == 0
        I1 = 0.5*ones(size(Itest,1),size(Itest,2));
    else
        I1 = 128*ones(size(Itest,1),size(Itest,2));
    end
    I1(cropParam(3):cropParam(4),cropParam(1):cropParam(2)) =...
        Itest(cropParam(3):cropParam(4),cropParam(1):cropParam(2));
    title_array = strcat('Cropped Marked Image (alpha=', num2str(alpha), ')')
    if flag8 == 0
        figure(5), imshow(I1), title(title_array)
    else
        figure(5), imshow(uint8(I1)), title(title_array)
    end
    Itest =I1;
    clear I1
 end
 [M,N] = size(Itest); % final dimensions after cropping
%*************************************************************************
%*******************************PROCESSING********************************
%------------------------------DENORMALIZING--------------------------------
if flag8 == 0
    if flagNorm == 1
        load C:\matlabR12\work\n
        Itest = 1/2 + tan(pi*(Itest-1/2))/n;
    end
```

```
    Itest = 255*Itest; % bring to range [0,255]
end
%-------------------------ORIGINAL IMAGE DCT--------------------------------
load C:\matlabR12\work\dctI
%--------------------------TEST IMAGE DCT-----------------------------------
T = dctmtx(8);
dctItest = blkproc(Itest,[8 8],'P1*x*P2',T,T');
clear Itest
%----------------------RECOVERED WATERMARK DCT (EXTRACTED)-----------------
dctWr = extract8(dctI,dctItest,Mw,Nw,index,indexGr,alpha,start,length);
clear dctI
clear dctItest
%---------------IDCT ON RECOVERED WATERMARK COEFFICIENTS-------------------
Wr = blkproc(dctWr,[8 8],'P1*x*P2',T',T); %recovered watermark
Wr = uint8(round(Wr));
clear dctWr
%--------------------------------BER-----------------------------------------
ber = BER(W,Wr);
fprintf(1,'BER(bits per pixel)=%1.4f\n',ber);
bermod = BERmod(W,Wr);
fprintf(1,'BERmod(bits per pixel with error)=%1.4f\n',bermod);
%--------------------------------rho-----------------------------------------
rho = corCoef(W,Wr);
fprintf(1,'rho=%1.4f\n',rho);
%--------------------------------XCORR---------------------------------------
% R = xcorr2(double(W),double(Wr));
% max_R = max(max(R));
% R = R/max_R;
%*****************************************************************************
%*****************************DISPLAY*****************************************
% if select == 0
%          title_array  =  strcat('Wm  recovered  from  marked  Image  (alpha=',
num2str(alpha), ')');
% elseif select == 1
%         title_array  =  strcat('Wm  recovered  from  quantized  Image  (alpha=',
num2str(alpha), ')');
% end


figure(11)
imagesc(Wr,[0          255]),          colormap(gray),          title('Recovered
Watermark')%title(title_array)
```

```
% figure(12)
% title_array = strcat('Xcor between W,Wr (a=', num2str(a), ')');
% mesh(R);
```

```
function [I] = imageSelectionC

%*************************************************************************
% Ioannis Retsas
% August, 2001
% LAST MODIFICATION: February 4, 2002
% FUNCTION: imageSelectionC
% INPUT: -
% DESCRIPTION: A routine that allows the user to select an image from the
% gallery.
% RETURNS: The image to be processed in grayscale uint8 form.
% NOTE:1. "fishingboat" and "pentagon" are already grayscale (no need for rgb2gray)
%         2. All the artificial images are saved in the corresponding files (imageB,
%         imageSB, imageR, imageU) with he same variable name "image".
%         3. The images must have dimensions that are multiple of 8 to be processed
%                by the framework; therefore "arctichare" and "newyork" are accordingly
%         modified.
%*************************************************************************

disp('Select image from gallery; Use...')
fprintf(1,'1 for Lena;\n2 for peppers;\n3 for fishing boat;\n');
fprintf(1,'4 for arctic hare;\n5 for New York;\n6 for pentagon;\n');
fprintf(1,'7 for imageB (blocks);\n8 for imageSB (small blocks);\n9 for imageR (random);\n');
file = input('10 for imageU (uniform);\n');
disp('Processing Data...')

if file == 1
    I = imread('lena7','bmp');
    I = rgb2gray(I);
elseif file == 2
    I = imread('peppers','bmp');
    I = rgb2gray(I);
elseif file == 3
    I = imread('fishingboat','bmp');
elseif file == 4
    I = imread('arctichare','bmp');
    I = rgb2gray(I);
```

```
    I = I(:,1:592);
elseif file == 5
    I = imread('newyork','bmp');
    I = rgb2gray(I);
    I = I(1:512,1:512);
elseif file == 6
    I = imread('pentagon','bmp');
elseif file == 7
    load C:\NINI\Thesis\Images\imageB
    I = image;
elseif file == 8
    load C:\NINI\Thesis\Images\imageSB
    I = image;
elseif file == 9
    load C:\NINI\Thesis\Images\imageR
    I = image;
elseif file == 10
    load C:\NINI\Thesis\Images\imageB
    I = image;
end
```

```
function [Ws, Key] = keyingC(W)

%*************************************************************************
% Ioannis Retsas
% Aug 31, 2001
% LAST MODIFICATION:February 14, 2002
% FUNCTION: keyingC
% INPUT: A matrix W of class uint8.
% DESCRIPTION: ...
% RETURNS: The keyed matrix Wk (uint8), and the Key used.
%*************************************************************************
[Mw,Nw] = size(W);
% W = uint8(round(W)); % if you include this you get an error if the input is
uint8
Key = uint8(round(rand(Mw,Nw,8)));
Wp = bitPlanes(W,8); % the watermark decomposed into planes
for k = 1:8
    Wk(:,:,k) = xor(Key(:,:,k),Wp(:,:,k)); % each plane is now coded with the
Key
end
Wk = ibitPlanes(Wk); % the keyed watermark is reassembled

Wk = uint8(Wk);
```

```
function [W] = WmTypeC(Mw,Nw)

%************************************************************************
% Ioannis Retsas
% Aug, 2001
% LAST MODIFICATION: February 4, 2002
% FUNCTION: WmTypeC
% INPUT: The dimensions Mw,Nw, of the watermark.
% DESCRIPTION: A function that accepts the watermark's dimensions, takes you
% through a watermark selection process, and...
% RETURNS: A grayscale watermark W.
% NOTE: If the NPS logo is selected, the size is by default 64x64 (not adjustable)
% and the input values are ignored.
%************************************************************************

fprintf('Select one of the watermarks. Use:\n')
selection = input('1 stripes;\n2 for NPS logo;\n3 for random grayscale watermark\n');
while (selection ~= 1)&(selection ~= 2)&(selection ~= 3)
    selection = input('Your choise should be one of 1,2 or 3; Try again:\n');
end
disp('Processing Data...')
if selection == 1
    W = stripes(Mw,Nw);
elseif selection == 2
    W = NPS;
elseif selection == 3
    W = randWm(Mw,Nw);
end
```

```
function [dctIo] = embed8(dctI,dctW,alpha,start,length)

%************************************************************************
% Ioannis Retsas
% Nov 15, 2001
% LAST MODIFICATION:-
% FUNCTION: embed8
% INPUT: The matrix dctI which will be marked, the matrix dctW which will
% mark dctI, the weighing factor alpha, the coefficient in each 8x8 block
% where the embedding starts, the number of coefficients that are embedded
% in each 8x8 block.
% DESCRIPTION: For each 8x8 block we calculate the CIPF (depending on each
% distance from the CI). The dctI is reshaped to 8x8x(M*N/64) (we sweep dctI
% left to right and top to bottom). For each block we calculate its CF & the PC
% (= E*RIPF)
% and we sort the blocks according to descending PCs. The watermark
coefficients
% are sorted by magnitude and divided into [length] number of groups. The
% elements of each group with the same index ([length] in total) form a set of
% embedding coefficients. Now each set is embedded into each 8x8 block starting
% from the [start]th coefficient. Finally we reshape the marked matrix to its
% original dimensions.
% RETURNS: A matrix dctIo with the marked coefficients.
% CAUTION:It is required that [length] divides exactly (Mw*Nw) and that
(Mw*Nw/length)
% is equal or smaller than the number of 8x8 image blocks.
%************************************************************************
[M,N] = size(dctI);
[Mw,Nw] = size(dctW);

% we mark each 8x8 block with its Eucledean distance from the center
% r(x,y) is the distance of the center of block (x,y) from the center of the
image
for m=1:8:M
    for n=1:8:N
        r(fix(m/8)+1,fix(n/8)+1) = (((m+3)-M/2)^2 + ((n+3)-N/2)^2)^(1/2);
    end
end

% reshaping the matrix from MxN to 8x8x(M*N/64)
```

```
% with this technique we sweep the matrix row-wise (left to right - top to
bottom)
k = 1;
for i = 1:8:M
    B(:,:,k:k+N/8-1) = reshape(dctI(i:i+7,:),8,8,N/8); % B is 8x8x4!!!
    k = k+N/8;
end


% we similarly (row-wise) reshape the matrix r with the distances
r_line = reshape(r',size(r,1)*size(r,2),1);


% we calculate for each block the CIPF (Center of Interest Proximity Factor)
rmax = max(max(r));
CIPF =  -1/pi*atan(15*(r/rmax-2/3))+1/2;


weight = []; % this is the vector that will accomodate the different weight
that is used for each coefficient
for i = 1:63
    weight = [weight i];
end


% F(i) is the CF calculated for each block i
for i = 1:size(B,3)
    B2 = B(:,:,i);
    V = zigzag(abs(B2));
    F(i) = weight*V(2:64)';
    % for each block we determine a Priority Coefficient PC which is the CF
    % weighted by the CIPF
    PC(i) = F(i)*CIPF(i);
end
PC = PC/max(F); % normalization


[varB(size(B,3):-1:1), index(size(B,3):-1:1)] = sort(PC);
B(:,:,:) = B(:,:,index); % B contains the 8x8 blocks sorted by descending order
of PC
save C:\matlabR12\work\index index


% sorting of the dct coefficients of the watermark by descending order of
magitude
[x, index2(Mw*Nw:-1:1)] = sort(abs(dctW(:)));
dctW = dctW(index2); % this way we avoid changing the values to positive after
sorting by var
```

```
% group the dct coefficients of the watermark in [length] groups
for i = 1:length
    gr(:,i) = dctW((i-1)*Mw*Nw/length+1:i*Mw*Nw/length)';
    indexGr(:,i) = index2((i-1)*Mw*Nw/length+1:i*Mw*Nw/length)';
end
save C:\matlabR12\work\indexGr indexGr
% embedding
for i = 1:size(gr,1)
    V = zigzag(B(:,:,i)); % V is a row vector that contains the elements of an 8x8 block aligned in zz fashion.
    V(start:start+length-1) = V(start:start+length-1) + alpha*gr(i,:);
    B(:,:,i) = zzRvs(V);
end


B(:,:,index) = B(:,:,:); % desorting the 8x8 dct blocks of the image to get their original order
k = 1;
for i = 1:8:M
    dctIo(i:i+7,:) = reshape(B(:,:,k:k+N/8-1),8,N); % contains the marked dct coefs of the image
    k = k+N/8;
end
```

```
function dctWr=extract8(dctI,dctIm,Mw,Nw,index,indexGr,alpha,start,length)


%************************************************************************
% Ioannis Retsas
% Sep 19, 2001
% LAST MODIFICATION:Nov 15, 2001
% FUNCTION: extract8
% INPUT: The matrix dctI, the marked matrix dctIm, the watermark dimensions Mw,
% Nw, the vectors index and indexGr from embed8, alpha, start and length.
% DESCRIPTION: We calculate the difference between the two input matrices.
% The dctDif is reshaped to 8x8xXXXX. The start to start+length-1 coeffs are
selected and
% are put back together to make up the watermark.
% RETURNS: The dct coeffs of the retrieved watermark.
%************************************************************************
[M,N] = size(dctI);
dctDif = dctIm - dctI;
k = 1;
for i = 1:8:M % reshape
    B(:,:,k:k+N/8-1) = reshape(dctDif(i:i+7,:),8,8,N/8); % B is 8x8x4!!!
    k = k+N/8;
end
B = B(:,:,index); % sorting using index (:,:,:)
embeddingSetsNumber = Mw*Nw/length;
for i = 1 : embeddingSetsNumber
    V = zigzag(B(:,:,i)); % V is a row vector that contains the elements of an
8x8 block aligned in zz fashion.
    dctWr(i,:) = V(start:start+length-1)/alpha;
    B(:,:,i) = zzRvs(V);
end



% k = 1; % counts the number of blocks that are being embedded.
% q = 1; % counts the 4 coefficients that are embedded in each block.
% for i = 1 : Mw*Nw
%       V = zigzag(B(:,:,k)); % V is a row vector that contains an 8x8 block
aligned in zz fashion.
%         dctWr(i) = V(6+q)/alpha; % embedding on the 7th,8th,9th and 10th
coefficients.
%     q = q+1;
%     if q == 5
```

```
%         q = 1;
%         k = k+1;
%     end
% end

dctWr(indexGr(:)) = dctWr(:); % desorting the dctW coefficients using indexNew
dctWr = reshape(dctWr,Mw,Nw);
```

```
function [Iq] = qFunc(I,q_jpeg)

%************************************************************************
% Ioannis Retsas
% February 11, 2002
% LAST MODIFICATION:-
% FUNCTION: qFunc
% INPUT: An image in the range [0 255], and a quality factor [1 100].
% DESCRIPTION: Performs 8x8 block quantization on I, using the standard JPEG
% luminance quantization table.
% RETURNS: The quantized image Iq (double [0 255] - may need to be
% transformed to uint8 8in order to be displayed).
%************************************************************************
I = double(I);
T = dctmtx(8);
dctI = blkproc(I,[8 8],'P1*x*P2',T,T');
Q = stdJPEGQ;
if q_jpeg <= 50
    q = 50/q_jpeg;
    Q = q*Q;
elseif (50 < q_jpeg) &  (q_jpeg <= 99)
    q = 2-(2*q_jpeg)/100;
    Q = q*Q;
else
    Q = ones(8);
end
dctI = blkproc(dctI,[8 8],'x./P1',Q); % as above (using of one of the Q tables)
dctI = round(dctI);
dctI = blkproc(dctI,[8 8],'x.*P1',Q);
Iq = blkproc(dctI,[8 8],'P1*x*P2',T',T); % image after quantization process
```

```
function [dctI] = quantFunc(dctI)

%************************************************************************
% Ioannis Retsas
% Aug, 2001
% LAST MODIFICATION: Oct 10, 2001
% FUNCTION: quantFunc
% INPUT: A matrix dctI (of DCT coefficients).
% DESCRIPTION: Performs 8x8 block quantization on matrix M, using one of the
% offered quantization matrices.
% RETURNS: The quantized matrix dctI (the quantized DCT coefficients).
%************************************************************************

fprintf('Select Quantization table; Press...\n');
QSelection = input('1 for binary table;\n2 for default JPEG table;\n3 for Image
Alchemy, Handmade Software Inc. table;\n');
while (QSelection ~= 1)&(QSelection ~= 2)&(QSelection ~= 3)
    QSelection = input('Your choise should be one of 1,2 or 3; Try again:\n');
end

if QSelection == 1
    comprRatio = input('Enter the compression ratio (x/64) for the DCT:\n');
    Q = binaryQ(comprRatio); % we assign to Q the values of a Quantization
table
elseif QSelection == 2
    Q = stdJPEGQ;
elseif QSelection == 3
    Q = IAHSIncQ;
end
if (QSelection == 2) | (QSelection == 3)
    q_jpeg = input('Type the value of the compression factor q_jpeg\n(default
value(%): 50):\n');
    while (q_jpeg < 1) | (q_jpeg > 100)
        q_jpeg = input('Your choise should be an integer in the range [1, 100];
Try again:\n');
    end
    if q_jpeg <= 50
        q = 50/q_jpeg;
        Q = q*Q;
    elseif (50 < q_jpeg) &  (q_jpeg <= 99)
```

```
        q = 2-(2*q_jpeg)/100;
        Q = q*Q;
    else
        Q = ones(8);
    end
end
disp('Processing Data...')
if QSelection == 1
    dctI = blkproc(dctI,[8 8],'P1.*x',Q); % dctI=dctImq, are the quantized coefs.
else
    dctI = blkproc(dctI,[8 8],'x./P1',Q); % as above (using of one of the Q tables)
    dctI = round(dctI);
    dctI = blkproc(dctI,[8 8],'x.*P1',Q);
end
```

```
function [q] = stdJPEGQ

%************************************************************************
% Ioannis Retsas
% Aug, 2001
% FUNCTION: stdJPEGQ
% INPUT: -
% DESCRIPTION: -
% RETURNS: The default JPEG quantization table.
%************************************************************************

q = [16  11  10  16  24  40  51  61;
     12  12  14  19  26  58  60  55;
     14  13  16  24  40  57  69  56;
     14  17  22  29  51  87  80  62;
     18  22  37  56  68 109 103  77;
     24  35  55  64  81 104 113  92;
     49  64  78  87 103 121 120 101;
     72  92  95  98 112 100 103  99];
```

```
function [R] = BER(W, Wr)

%************************************************************************
% Ioannis Retsas
% Nov 14, 2001
% LAST MODIFICATION:-
% FUNCTION: BER
% INPUT: Two equally sized matrices W, Wr (uint8).
% DESCRIPTION: Calculates the Bit Error Rate (in error bits per pixel).
% RETURNS: The Bit Error Rate, R.
%************************************************************************

[Mw,Nw] = size(W);
[Mwr,Nwr] = size(Wr);
if (Mw ~= Mwr) | (Nw ~= Nwr)
    display('The two iputs do not have the same size; rho will not be
calculated')
    return
end
Wb=bitPlanes(W,8);
Wrb=bitPlanes(Wr,8);
sum=0;
for i=1:Mw
    for j=1:Nw
        for k=1:8
            if Wb(i,j,k) ~= Wrb(i,j,k)
                sum = sum+1;
            end
        end
    end
end
R = sum/(Mw*Nw);
```

```
function [R] = BERmod(W, Wr)

%************************************************************************
% Ioannis Retsas
% Nov 14, 2001
% LAST MODIFICATION:-
% FUNCTION: BERmod
% INPUT: Two equally sized matrices W, Wr (uint8).
% DESCRIPTION: Calculates the Bit Error Rate (in error bits per pixel of error).
% RETURNS: The Bit Error Rate, R.
%************************************************************************

[Mw,Nw] = size(W);
[Mwr,Nwr] = size(Wr);
if (Mw ~= Mwr) | (Nw ~= Nwr)
    display('The two iputs do not have the same size; rho will not be calculated')
    return
end
Wb=bitPlanes(W,8);
Wrb=bitPlanes(Wr,8);
sum = 0; sum1 = 0; % counters
for i=1:Mw
    for j=1:Nw
        flag = 0; % the flag is set to 0 for each new pixel
        for k=1:8
            if Wb(i,j,k) ~= Wrb(i,j,k)
                flag = 1; % the flag is set to 1 when an error occurs in a pixel
                sum = sum+1;
            end
        end
        if flag == 1
            sum1 = sum1+1;
        end
    end
end
R = sum/sum1;
```

```
function [Ip] = bitPlanes(I,type)

%************************************************************************
% Ioannis Retsas
% Aug 17, 2001
% LAST MODIFICATION: Sep 5, 2001
% FUNCTION: bitPlanes
% INPUT: A grayscale Image, the data type (8 for uint8, 16 for uint16...)
% DESCRIPTION: Receives a grayscale -uint8- image as an input. For each pixel
% of the image gets the binary representation of its value. Creates a set of
% binary image planes, each containing one bit per pixel. Each plane contains
% bits of the same significance.
% RETURNS: A binary matrix where the first two dimensions are the actual
% dimensions of the input image, while the third dimension represents the
% different planes, each containing equally significant bits of the binary
% representation of the value of each pixel. Plane 1 (k=1) contains the most
% significant bits.
%************************************************************************

[M,N] = size(I);
% I = uint8(round(I)); % PROSOXH
for i = 1:M
    for j = 1:N
        for k = type:-1:1
            Ip(i,j,type+1-k) = bitget(I(i,j),k);
        end
    end
end
```

```
function [rho] = corCoef(W, Wr)

%************************************************************************
% Ioannis Retsas
% Nov 14, 2001
% LAST MODIFICATION:-
% FUNCTION: corCoef
% INPUT: Two equally sized matrices W, Wr.
% DESCRIPTION: After substracting the mean, calculates the cross correlation
% of the two matrices for the instant that the two matrices are aligned.
% RETURNS: rho (the cross correlation of the two matrices for the instant
% that the two matrices are aligned).
%************************************************************************

[Mw,Nw] = size(W);
[Mwr,Nwr] = size(Wr);
if (Mw ~= Mwr) | (Nw ~= Nwr)
    display('The two inputs do not have the same size; The program is
terminated')
    return
end
m_r = mean(mean(double(Wr)));
m = mean(mean(double(W)));
Wr = double(Wr)-m_r;
W = double(W)-m;
sum_x = 0; sum = 0; sum_r = 0; % counters
for i = 1:Mw
    for j=1:Nw
        sum_x = sum_x + W(i,j)*Wr(i,j);
        sum = sum + W(i,j)^2;
        sum_r = sum_r + Wr(i,j)^2;
    end
end
rho = sum_x/(sum*sum_r)^(1/2);
```

```
function [I] = ibitPlanes(Ip)

%************************************************************************
% Ioannis Retsas
% Aug 17, 2001
% LAST MODIFICATION: Sep 4, 2001
% FUNCTION: ibitPlanes
% INPUT: A binary matrix where the first two dimensions are the actual
% dimensions of the input image, while the third dimension represents the
% different planes, each containing equally significant bits of the binary
% representation of the value of each pixel. Plane 1 (k=1) contains the most
% significant bits.
% DESCRIPTION: Performs the reverse process of bitPlanes function.
% Receives an image that has been decomposed into binary planes of equally
% significant bits, and returns the original image.
% RETURNS: A grayscale Image.
%************************************************************************
% Ip = uint8(round(Ip)); % PROSOXH
[L,M,N] = size(Ip);

for i = 1:L
    for j = 1:M
        C = Ip(i,j,1);
        for k = 1:N-1
            C = bitshift(C,1);
            C = bitor(C,Ip(i,j,k+1));
        end
        I(i,j) = C;
    end
end
```

```
function s=SNR(M,Md)
%************************************************************************
% Ioannis Retsas
% Nov 16, 2001
% LAST MODIFICATION:-
% FUNCTION: SNR
% INPUT: The original matrix M, and the distorted matrix Md, expressed in a
% [0 255] scale.
% DESCRIPTION: We calcuated the SNR based on the formula:
% SNR = 10log(sigma^2/mse), where mse = E[(x-xd)^2] = (1/MN)SUM SUM (x-xd)^2
% is the mean square error, as presented in introduction to Data Compression by
% Khalid Sayood (2nd edition), 2000.
% RETURNS: The SNR, s.
%************************************************************************
M = double(M); Md = double(Md);
mean_M = mean(mean(M));
sigmaSquare = 0; mse = 0;
% calculation of sigma, mse
[K,L] = size(M);
for i = 1 : K
    for j = 1 : L
        sigmaSquare = sigmaSquare + (M(i,j)-mean_M)^2;
        mse = mse + (M(i,j) - Md(i,j))^2;
    end
end
sigmaSquare = sigmaSquare/(K*L);
mse = mse/(K*L);

s = 10*log10(sigmaSquare/mse); %
```

```
function [V] = zigzag(Mi)

%********************************************************************************
% Ioannis Retsas
% September 12, 2001
% LAST MODIFIED: -
% FUNCTION: zigzag(Mi)
% INPUT: An input matrix Mi.
% DESCRIPTION: -
% RETURNS: A row vector V that contains the elements arranged in zigzag
% fashion.
%********************************************************************************
A = [ 1  2  6  7 15 16 28 29;
      3  5  8 14 17 27 30 43;
      4  9 13 18 26 31 42 44;
     10 12 19 25 32 41 45 54;
     11 20 24 33 40 46 53 55;
     21 23 34 39 47 52 56 61;
     22 35 38 48 51 57 60 62;
     36 37 49 50 58 59 63 64];

 V(A(:)) = Mi(:);
```

```
function [Mo] = zzRvs(V)

%*************************************************************************
% Ioannis Retsas
% September 12, 2001
% LAST MODIFIED: -
% FUNCTION: zzRvs(Mi)
% INPUT: An input vector V.
% DESCRIPTION: -
% RETURNS: A matrix Mo that has the elements of V arranged in a zigzag fashion.
%*************************************************************************
A = [ 1  2  6  7 15 16 28 29;
      3  5  8 14 17 27 30 43;
      4  9 13 18 26 31 42 44;
     10 12 19 25 32 41 45 54;
     11 20 24 33 40 46 53 55;
     21 23 34 39 47 52 56 61;
     22 35 38 48 51 57 60 62;
     36 37 49 50 58 59 63 64];

Mo = V(A(:));
Mo = reshape(Mo,8,8);
```

```
function [W] = NPS

%*************************************************************************
% Ioannis Retsas
% Nov 19, 2001
% FUNCTION: NPS
% INPUT: -
% DESCRIPTION:
% RETURNS: Returns a gray scale watermark W, 64x64, with the NPS logo
% comprised by blocks of different (random) grayscale level;
% REMARK: Each element of the matrix is an uint8. Whether it will be color
% or gray depends on the function you are using.
% imagesc() gives the colored representation, while
% imagesc(), colormap(gray) gives the gray scale one.
%*************************************************************************
% background
W = round(255*rand(64,64));
W(11:50,:) = 200*ones(40,64);
% N
W(17:42,5:8) = round(255*rand(26,4));
W(19:20,9) = round(255*rand(2,1));
W(21:22,9:10) = round(255*rand(2,2));
W(23:24,9:11) = round(255*rand(2,3));
W(25:26,9:12) = round(255*rand(2,4));
W(27:28,10:13) = round(255*rand(2,4));
W(29:30,11:14) = round(255*rand(2,4));
W(31:32,12:15) = round(255*rand(2,4));
W(33:34,13:16) = round(255*rand(2,4));
W(35:36,14:16) = round(255*rand(2,3));
W(37:38,15:16) = round(255*rand(2,2));
W(39:40,16) = round(255*rand(2,1));
W(17:42,17:20) = round(255*rand(26,4));
% P
W(17:42,25:28) = round(255*rand(26,4));
W(17:20,29:32) = round(255*rand(4,4));
W(18:21,33:34) = round(255*rand(4,2));
W(20:23,35:36) = round(255*rand(4,2));
W(22:26,37:38) = round(255*rand(5,2));
W(25:28,35:36) = round(255*rand(4,2));
W(27:30,33:34) = round(255*rand(4,2));
```

```
W(28:31,29:32) = round(255*rand(4,4));
% S
W(19:22,55:56) = round(255*rand(4,2));
W(18:21,53:54) = round(255*rand(4,2));
W(17:20,49:52) = round(255*rand(4,4));
W(18:21,47:48) = round(255*rand(4,2));
W(20:23,45:46) = round(255*rand(4,2));
W(22:26,43:44) = round(255*rand(5,2));
W(25:28,45:46) = round(255*rand(4,2));
W(27:30,47:48) = round(255*rand(4,2));
W(28:31,49:52) = round(255*rand(4,4));
W(29:32,53:54) = round(255*rand(4,2));
W(31:34,55:56) = round(255*rand(4,2));
W(33:37,57:58) = round(255*rand(5,2));
W(36:39,55:56) = round(255*rand(4,2));
W(38:41,53:54) = round(255*rand(4,2));
W(39:42,49:52) = round(255*rand(4,4));
W(38:41,47:48) = round(255*rand(4,2));
W(36:39,45:46) = round(255*rand(4,2));

W = uint8(W);
```

```
function [W] = randWm(M,N)

%*************************************************************************
% Ioannis Retsas
% Nov 25, 2001
% FUNCTION: randWm
% INPUT: The dimensions M, N of the watermark.
% DESCRIPTION:
% RETURNS: Returns a gray scale random watermark W (MxN).
%*************************************************************************
W = uint8(round(255*rand(M,N)));
```

```
function [W] = stripes(M,N)

%************************************************************************
% Ioannis Retsas
% Aug, 2001
% FUNCTION: stripes
% INPUT: The dimensions M, N of the watermark.
% DESCRIPTION:
% RETURNS: Returns a gray scale watermark W; 11 vertical stripes with the
% the value of the gray scale be maximum in the middle stripe.
% REMARK: Each element of the matrix is an uint8. Whether it will be color
% or gray depends on the function you are using.
% imagesc() gives the colored representation, wlile
% imagesc(), colormap(gray) gives the gray scale one.
%************************************************************************

width = round(N/11);

W(1:M, 1:width) = 0 * ones(M,width);
W(1:M, width+1 : 2*width) = 50 * ones(M,width);
W(1:M, 2*width+1 : 3*width) = 100 * ones(M,width);
W(1:M, 3*width+1 : 4*width) = 150 * ones(M,width);
W(1:M, 4*width+1 : 5*width) = 200 * ones(M,width);
W(1:M, 5*width+1 : 6*width) = 250 * ones(M,width);
W(1:M, 6*width+1 : 7*width) = 225 * ones(M,width);
W(1:M, 7*width+1 : 8*width) = 175 * ones(M,width);
W(1:M, 8*width+1 : 9*width) = 125 * ones(M,width);
W(1:M, 9*width+1 : 10*width) = 75 * ones(M,width);
W(1:M, 10*width+1 : N) = 25 * ones(M, N-10*width);

W = uint8(W);
```

LIST OF REFERENCES

[1] F. A. P. Petitcolas, R. J. Anderson, M. G. Kuhn, "Information Hiding – A Survey", *Proceedings of the IEEE*, vol.87, no.7, pp.1062-1078, July 1999.

[2] D. J. Ryan, "Infosec and Infowar, Considerations for Military Intelligence", www.danjryan.com/MIntl.html.

[3] F. Hartung, M. Kutter, "Multimedia Watermarking Techniques", *Proceedings of the IEEE*, vol.87, no.7, pp.1079-1107, July 1999.

[4] I. J. Cox, M. L. Miller, J. A. Bloom, "Digital Watermarking", Morgan Kaufmann Publishers, 2002.

[5] C. P. Pfleeger, "Security in Computing", Prentice Hall PTR, 2nd edition, 2000.

[6] G. L. Friedman, "The Trustworthy Digital Camera: Restoring Credibility to the Photographic Image", *IEEE Transactions on Consumer Electronics*, vol.39, pp.905-910, October 1993.

[7] D. Kundur, D. Hatzinakos, "Digital Watermarking for Telltale Tamper Proofing and Authentication", *Proceedings of the IEEE,* vol.87, no.7, pp.1167-1180, July 1999.

[8] C. I. Podilchuk, E. J. Delp, "Digital Watermarking: Algorithm and Applications", *IEEE Signal Processing Magazine*, pp.33-46, July 2001.

[9] C. C. Langelaar, I. Setyawan, R. L. Lagendijk, "Watermarking Digital Image and Video Data", *IEEE Signal Processing Magazine*, pp.20-46, September 2000.

[10] W. Bender, D. Gruhl, N. Morimoto, "Techniques for Data Hiding", *Proceedings SPIE*, vol.2420, p.40, San Jose, CA, February 1995.

[11] N. Nikolaidis, I. Pitas, "Copyright Protection of Images using Robust Digital Signatures", *Proceedings ICASSP '96*, Atlanta, GA, May 1996.

[12] C. Langelaar, J. C. A. van der Lubbe, R. L. Lagendijk, "Robust Labeling Methods for Copy Protection of Images", *Proceedings in Electronic Imaging*, vol.3022, pp.298-309, San Jose, CA, February 1997.

[13] M. Kutter, F. Jordan, F. Bossen, "Digital Signature of Color Images using Amplitude Modulation", *Proceedings in Electronic Imaging*, San Jose, CA, February 1997.

[14] M. Barni, C. I. Podilchuk, F. Bartolini, E. J. Delp, "Watermark Embedding: Hiding a Signal within a Cover Image", *IEEE Communications Magazine*, pp.102-108, August 2001.

[15] M. Ramkumar, A. N. Akansu, A. A. Alatan, "On the Choice of Transforms for Data Hiding in Compressed Video", *IEEE International Conference on Acoustics, Speech, and Signal Processing 1999, Proceedings*, Vol.6 , pp.3049-3052, 1999.

[16] S. Kang, Y. Aoki, “Digital Image Watermarking by Fresnel Transform and its Robustness”, *International Conference on Image Processing, Proceedings 1999 (ICIP 99)*, vol.2, pp.221-225.

[17] J. J. K. O Ruanaidh, W. J. Dowling, F. M. Boland, "Phase Watermarking of Digital Images", Proceedings IEEE, International Conference of Image Processing, vol.III, pp.239-242, Lausanne, Switzerland, September 16-19, 1996.

[18] A. Herrigel, H. Petersen, J. O Ruanaidh, T. Pun, P. Shelby, "Copyright Techniques for Digital Images Based on Asymmetric Cryptographic Techniques", *presented at Workshop on Information Hiding*, Portland, Oregon, USA, April 1998.

[19] A. Herrigel, J. J. K. O Ruanaidh, H. Petersen, S. Pereira, T. Pun, "Secure Copyright Protection Techniques for Digital Images", *in Information Hiding (Lecture Notes in Computer Science, vol. 1525)*, D. Aucsmith, Ed. Berlin, Germany: Springer, 1998, pp.169-190.

[20] S. Pereira, J. J. K. O Ruanaidh, F. Deguillaume, G. Csurka, T. Pun, "Template-based Recovery of Fourier-based Watermarks using Log-polar and Log-log Maps", *Proceedings IEEE in Multimedia Systems 99, International Conference in Multimedia Computing and Systems*, Florence, Italy, June 7-11, 1999.

[21] J. J. K. O Ruanaidh, F. M. Boland, O. Sinnen, "Watermarking Digital Images for Copyright Protection", *Proceedings in Electronic Imaging and the Visual Arts 1996*, Florence, Italy, February 1996.

[22] I. J. Cox, J. Kilian, F. T. Leighton, T. Shamoon, “Secure Spread Spectrum Watermarking for Multimedia”, *IEEE Transactions on Image Processing*, vol.6, no.12, pp.1673-1687, December 1997.

[23] C. I. Podilchuk, W. Zeng, “Image-adaptive Watermarking using Visual Models”, *IEEE Journal on Selected Areas in Communications*, vol.16, no.4, pp.525-539, May 1998.

[24] C. I. Podilchuk, W. Zeng, “Perceptual Watermarking of Still Images”, *IEEE First Workshop on Multimedia Signal Processing*, pp.363-368, 1997.

[25] A. Piva, M. Barni, E. Bartolini, V. Cappellini, "A DCT-based Watermarking Recovering without resorting to the Uncorrupted Digital Image", *Proceedings IEEE, International Conference in Image Processing*, vol.1, p.520, Santa Barbara, CA, 1997.

[26] M. Barni, F. Bartolini, V. Cappellini, A. Lipi, A. Piva, "A DWT-based Technique for Spatio-Frequency Masking of Digital Signatures", *Proceedings SPIE/IS International Conferencein Security and Watermarking of Multimedia Contents*, vol.3657, pp.31-39, San Jose, January 25-27, 1999.

[27] K. Sayood, "Introduction to Data Compression", Morgan Kaufmann Publidhers, 2nd edition, 2000.

[28] J. Miano, "Compressed Image File Formats", Addison Wesley Longman, Inc, 1999.

[29] K. R Castleman, "Digital Image Processing", Prentice Hall, Inc, 1996.

[30] M. Ghanbari, "Video Coding: An Introduction to Standard Codecs", IEE Telecommunication Series 42, 1999.

[31] C.-T. Hsu, J.-L. Wu, "Hidden Digital Watermarks in Images"**,** *IEEE Transactions on Image Processing*, vol.8, no.1, pp. 58-68, January 1999.

[32] K. S. Shanmugan, A. M. Breipohl, "Random Signals: Detection Estimation and Data Analysis", John Wiley and Sons, Inc, 1988.

[33] R. B Wolfgang, C. I. Podilchuk, E. J. Delp, "Perceptual Watermarks for Digital Images and Video", *Proceedings of the IEEE,* vol.87, no.7, pp. 1108-1126, July 1999.

[34] C. I. Podilchuk, E. J. Delp, "Digital Watermarking: Algorithm and Applications", *IEEE Signal Processing Magazine*, pp.33-46, July 2001.

[35] Y.-P. Wang, M.-J. Chen, p.-Y. Cheng, "Robust Image Watermark With Wavelet Transform and Spread Spectrum Techniques", Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, vol.2 , pp. 1846 –1850, Asilomar, 2000.

[36] S. B. Wicker, "Error Control Systems for Digital Communication and Storage", Prentice-Hall Inc, 1995.

[37] I. Retsas, R. Pieper, R. Cristi, "Watermark Recovery with a DCT-based Scheme Employing Nonuniform Imbedding", *34th Southeastern Symposium on System Theory (SSST-02)*, Alabama, March 18-19, 2002.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California, 93943-5121

3. Chairman, Department of Electrical and Computer Engineering
   Monterey, California, 93943-5121

4. Chairman, Information Warfare Academic Group
   Monterey, California, 93943-5121

5. Prof. Ron Pieper
   Department of Electrical and Computer Engineering
   Monterey, California, 93943-5121

6. Prof. Roberto Cristi
   Department of Electrical and Computer Engineering
   Monterey, California, 93943-5121

7. Hellenic Navy General Staff
   Department B2, Stratopedo Papagou, Mesogeion 151,
   Holargos, 15500, Athens
   Greece

8. Lt Ioannis Retsas HN
   Eirinis 49-51, Ag. Paraskevi
   Athens, 15341
   Greece